PROBLEMS IN INVESTIGATION THEORY

Stephen John Balut

Library
Naval Postgraduate School
Monterey, California 93940

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

PROBLEMS IN INVESTIGATION THEORY

by

Stephen John Balut

Thesis Advisor: G.T. Howard

March 1973

Approved for public release; distribution unlimited.

T153377

Problems in Investigation Theory

by

Stephen John Balut
Lieutenant Commander, United States Navy
B.A., Kings College, 1960

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

from the

NAVAL POSTGRADUATE SCHOOL

Library
Naval Postgraduate School
Monterey, California 93940

ABSTRACT

Investigation theory treats discrete combinatorial optimization problems in which there are several objects passing through a region containing one or more investigators who are to investigate, according to some criteria, objects prior to their escape across a portion of the boundary of the region. In general, investigation times are sequence-dependent functions of the time investigation is initiated. This research treats problems with one investigator under the criteria of minimization of the number of objects to escape uninvestigated. Those problems for which optimal solutions can be efficiently obtained are identified and algorithms developed. For the general problem, heuristic solution methods are suggested and evaluated through comparison of results obtained with optimal solutions. An analysis is presented for problems with uncertain investigation times and also for problems in which objects are not immediately available for investigation. Generalizations to more than one investigator and an alternate objective are discussed. The relationship between investigation and jobshop scheduling problems is illustrated throughout.

TABLE OF CONTENTS

I. INTRODUCTION TO INVESTIGATION THEORY ---------- 7

A. INTRODUCTION ---------- 7

B. CURRENT RESEARCH RELATED TO INVESTIGATION THEORY ---------- 9

C. RESEARCH OBJECTIVES ---------- 11

II. PROBLEM DESCRIPTION ---------- 14

A. GENERAL DESCRIPTION ---------- 14

B. MATHEMATICAL DESCRIPTION ---------- 15

C. DISCUSSION ---------- 17

D. SUMMARY OF PROBLEM TREATMENT ---------- 22

III. SPECIFIED ORDERING ---------- 24

A. PRELIMINARY REMARKS ---------- 24

B. DYNAMIC PROGRAMMING FORMULATION AND SOLUTION OF PROBLEMS HAVING A SPECIFIED ORDERING ---------- 25

1. Dynamic Programming Formulation of Problems With One Investigator on the Border ---------- 26

2. Graphical Solution for the Problem With One Investigator on the Border ----- 29

3. Multiple Investigators on the Border ---- 30

4. Example and Solution ---------- 32

5. Arbitrary Orderings ---------- 36

6. Further Generalizations ---------- 37

C. A MATRIX ALGORITHM FOR PROBLEMS WITH A SPECIFIED ORDERING ---------- 39

1. The Algorithm ---------- 40

2. Example ---------- 41

3. Proof of Optimality ---------- 44

4. Discussion ---------- 46

IV. EVALUATION OF HEURISTIC SOLUTION METHODS -------- 47

A. INTRODUCTION ---------- 47

B. EVALUATION OF THREE HEURISTIC SOLUTION METHODS ---------- 49

1. Same Target Speeds ---------- 51

2. Different Target Speeds ---------- 54

C. SYNOPSIS OF ALGORITHMS USED FOR OBTAINING OPTIMAL SOLUTIONS ---------- 55

1. Preliminary Remarks ---------- 55

2. Application of Branch and Bound to Investigation Problems ---------- 56

3. Framework for Algorithms for Solving Investigation Problems ---------- 60

4. Difficulties Associated with Selection of Branching and Bounding Rules --------- 61

D. PROBLEMS WHICH CAN BE SOLVED USING THE ALGORITHM ---------- 3

1. Discussion ---------- 3

2. Determination of Suitability of a Formulation for Application of the Algorithm ---------- 64

3. Types of Motion Limiting the Applicability of the Algorithm --------- 69

E. AN ALGORITHM FOR TARGETS WITH DIFFERENT COURSES AND SPEEDS ---------- 71

1. Problem Description ---------- 72

2. Flow Diagram ---------- 73

3. Sample Problem and Solution ---------- 86

4. Remarks ---------- 89

V. UNCERTAIN INVESTIGATION TIMES ---------- 91

A. INTRODUCTORY REMARKS ---------- 91

B. SCHEDULING WITH DUE-DATES AND UNCERTAIN SET-UP AND PROCESSING TIMES ---------- 93

1. Integer Programming Formulation -------- 94

2. Chance Constrained Formulation --------- 94

3. Development of the Deterministic Equivalent ---------- 97

4. The Algorithm ---------- 100

5. Example ---------- 101

6. Proof of Optimality ---------- 102

C. REMARKS ---------- 105

VI. NON-ZERO RELEASE TIMES ---------- 106

A. INTRODUCTION ---------- 106

1. Problem Statement ---------- 108

2. Separability ---------- 110

B. DUE-DATE SEQUENCES ---------- 114

1. Partial DD Orderings ---------- 114

2. Due-Date Algorithm ---------- 119

C. RELEASE-TIME SEQUENCES ---------- 125

1. Partial R Orderings ---------- 126

2. Release-Time Algorithm ---------- 129

D. GENERAL SEQUENCES ---------- 135

1. Special Cases ---------- 135

2. Combinatorial Accelerations ---------- 137

a. Job Ranges ---------- 138

b. Complete Specification of Z -------- 140

3. Remarks ---------- 143

VII. GENERALIZATIONS ---------- 145

A. INTRODUCTION ---------- 145

B. THE VALUE PROBLEM ---------- 145

1. Problem Statement ---------- 146

2. Summary of Results Applicable to the Value Problem ---------- 146

3. Possible Solution Methods ---------- 149

C. PROBLEMS WITH MORE THAN ONE INVESTIGATOR ---- 151

1. Integer Programming Formulation --------- 152

2. Discussion ---------- 152

D. OTHER PROBLEMS ---------- 155

LIST OF REFERENCES ---------- 157

INITIAL DISTRIBUTION LIST ---------- 161

FORM DD 1473 ---------- 162

I. INTRODUCTION TO INVESTIGATION THEORY

A. INTRODUCTION

This research deals with problems in an area called Investigation Theory [Refs. 8 , 36]. In these problems, an investigator and several objects, called targets, are located in a region and some of the targets are proceeding toward a border of the region. The investigator is to investigate targets prior to their crossing the border in order to optimize with respect to some prescribed objective.

Apart from the objective, such problems resemble the traveling salesman problem and certain classes of job-shop scheduling problems. However, the feature of investigation problems which sets them apart from other classes of problems appearing in the literature is the criteria of optimization. Although many specific criteria can be developed, all are concerned with completing investigations prior to target escape across the border. This research represents an analysis of investigation problems under the objective of minimization of the number of targets to escape across the border uninvestigated.

The author was introduced to Investigation Theory while attached to the Operations Evaluation Group at the Center for Naval Analysis. As a result of analysis of several on-going military operations, a deficiency in existing theory was recognized. The particular problem which illuminated

this deficiency was related to the Markettime operation which was being conducted along the coast of South Vietnam.

In the early stages of the Vietnam conflict a principal means of logistic support for the guerilla forces operating in South Vietnam was via the sea. Ships of various sizes would rendezvous at predetermined locations on the beaches and offload needed supplies and food. In an effort to sever or at least curtail this source of supply, South Vietnamese and allied forces divided the coastal waters into regions and assigned patrol craft to each region. The search of coastal waters was conducted primarily by aircraft, and detected surface craft were intercepted and inspected by surface patrol craft.

Procedures for carrying out the air search for possible infiltrators were quickly obtained through application of methods previously developed and embodied in the theory of search and detection. The responsibility for investigating those surface craft detected to determine whether they were carriers of enemy supplies rested with the surface patrol craft assigned to the regions. In many instances the number of contacts to be investigated exceeded the capabilities of the surface patrol craft. Since there did not exist a unified body of knowledge interfacing with and complementing the theory of search and detection for use in developing procedures for the patrol craft commanders to follow in such situations, they simply used their best judgement in attempting to carry out their assigned missions.

Analysts considering this problem found themselves unable to specify optimal policies for conducting such investigations, and in fact, were not even able to suggest one usable rule of thumb which was known to be good.

Recognizing this deficiency, Dr. J. A. Neuendorffer submitted a request for analysis [Ref. 8] in which he used the term "investigation problem" to describe such situations and dubbed that body of knowledge to be developed to treat them as "Investigation Theory." Except for a literature search [Ref. 36], apparently no research was directed toward this area until 1970 when the research being reported here was begun.

This research does not treat any problem related to the development of sets of strategies for use by opposing forces, however, the results obtained would perhaps be of use to game theorists in any such endeavor.

B. CURRENT RESEARCH RELATED TO INVESTIGATION THEORY

The investigation problem was identified and the term "investigation theory" defined in Ref. 8 . Literature searches for related research are contained in Refs. 35 and 36.

In Ref. 8 it is suggested that the development of investigation theory should begin through consideration of problems in which all targets have speed zero. The similarity between such problems and the traveling salesman problem suggested application of existing solution methods for the

latter problem. The first solution method designed specifically for an investigation problem is presented in Ref. 35, in which the zero target speed problem is solved using a modification of the branch and bound algorithm of Little, et al. [Ref. 27].

Solution methods developed for various types of target motion are contained in Refs. 1, 2 , 35, and 41, all of which are incorporated into this dissertation.

The investigation problem treated here is closely related to the jobshop scheduling problem in which the objective is to minimize the number of late jobs. This problem is described by letting p_i be the combined set-up and processing time for job i and d_i the due-date. Let decision variables x_i=1 if job i is included in the schedule and 0 otherwise, for i = 1,2,...,n. The problem is to

$$\text{maximize} \quad \sum_{i=1}^{n} x_i$$

$$\text{subject to} \quad \sum_{j=1}^{i} p_j x_j \leq d_i \qquad i = 1,2,\ldots,n.$$

For the case where the combined set-up and processing times are deterministic and sequence independent, Jackson [38] proves the existence of an optimal schedule in which all jobs are ordered according to non-decreasing due-dates. In Ref. 32 a very simple algorithm for obtaining optimal solutions for this problem is presented. Problems in which set-up or

processing times are sequence dependent are usually formulated as traveling salesman problems [Ref. 12].

Existing solution techniques for the traveling salesman problem are summarized in Ref. 4 , in which, after an analysis of computational results, it is recommended that dynamic programming [Ref. 21] be used for problems with 13 cities or less, and that branch and bound [Ref. 38] be used for the rest. A rather complete survey of branch and bound methods is contained in Ref. 26. Several applications of the branch and bound technique to scheduling problems are contained in Refs. 5 , 22, 27, and 31.

Other references are cited throughout the dissertation as their relation to the investigation problem becomes apparent.

C. RESEARCH OBJECTIVES

The broad objective of this research is an analysis of investigation problems with one investigator in which the objective is minimization of the number of targets to escape uninvestigated, with generalization, where possible, to related problems. The principal goal has been the development of practical techniques for obtaining optimal or near optimal solutions to the class of problems described, while at the same time developing a theoretical framework for understanding more general problems.

Specific objectives include the following:

1. Subdivide the class of problems, identifying those for which optimal solutions can be obtained relatively easily

through exploitation of special structure, and develop efficient algorithms for obtaining these optimal solutions.

2. For those problems whose structure does not lend itself to solution methods efficient enough for practical use, evaluate heuristic solution methods.

3. Determine the most general class of problems for which solutions can be expected to be obtained.

4. Provide a theoretical basis for extending this class through analysis of problem structure.

5. Develop an algorithm for obtaining optimal solutions to the most general problem treated in support of objectives 2 through 4.

It is hoped that the results obtained in this research will provide others with a foundation upon which to build a unified theory of investigation.

The results obtained are reported in the sections which follow. In Section II a mathematical description of the investigation problem is presented, followed by a discussion of its analytical characteristics and the relative merit of possible approaches. Section III identifies a class of problems with special structure for which optimal solution methods have been developed. Section IV deals with those problems for which methods yielding optimal solutions are computationally impractical. Methods of development and evaluation of heuristic solution techniques are suggested and illustrated by example. The algorithm developed in fulfillment of objective 5 is presented with sufficient

description to be of use to others who may wish to evaluate new or alternate non-optimal solution methods. Section V discusses the implications of uncertain investigation times and identifies a special class of problems for which an extremely efficient optimal solution method has been developed. In Section VI, problems in which all targets are not immediately available for investigation are considered. Section VII discusses generalizations of the problem and their applicability, and suggests possible avenues for future research.

II. PROBLEM DESCRIPTION

A. GENERAL DESCRIPTION

An Investigation problem is one in which there are several targets passing through a region containing a distinguished object, called the investigator, who is to investigate, according to some criteria, the targets prior to their escape across a portion of the boundary of the region, called the border. An optimal solution to such a problem is the specification of which targets will be investigated, and in what sequence, in order to optimize with respect to this criteria.

It is assumed that the following information is either implied or specified in an investigation problem formulation.

1. The meaning of Investigation.
2. The region in which targets are located and the subregion to which the motion of the investigator is restricted.
3. The motion characteristics of the investigator and targets.
4. The criteria or objective of the investigator.

For illustration, consider the example situation depicted in Figure 2.1. The region where the targets and investigator are located is planar and the investigator is restricted to remain in the square subregion. Target locations are represented by numbered dots and their motion by vectors emanating from their locations. The Investigator is in position 1. Suppose an investigation is considered complete when the location of a target is reached by the investigator

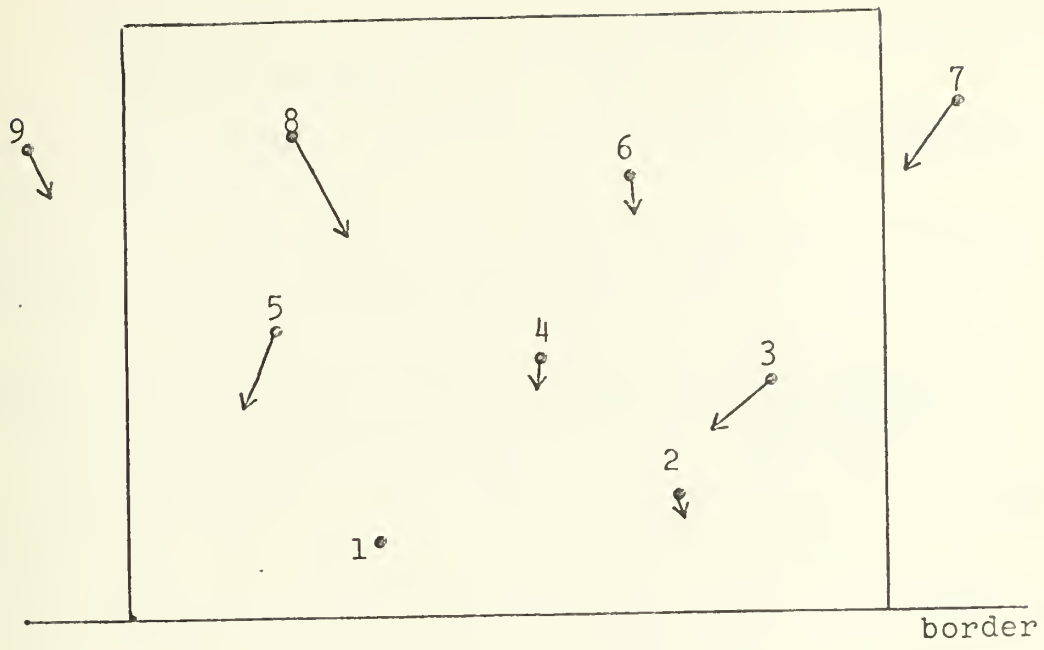


Figure 2.1

and let the objective be minimization of the number of targets to cross the border uninvestigated. An optimal solution is a path for the investigator to follow which contains the greatest number of targets.

B. MATHEMATICAL DESCRIPTION

The investigator is always designated as target number one. A solution to an investigation problem is represented by a permutation of target numbers indicating an order in which targets included are investigated. Square brackets are used to denote position in sequence in a permutation, thus the symbol [i] represents the target number which is in the i^{th} position in sequence in some permutation.

The following notation is used. Let

π = $(\pi_1,\pi_2,\ldots,\pi_m)$ be a permutation of m elements from the set $\{1,2,\ldots,n\}$, $m \leq n$, representing an order in which targets $\pi_1,\pi_2,\ldots,\pi_m$ are investigated.

a_i = the earliest time at which target i is available for investigation for i=2,...,n.

e_i = the escape time of target i if not investigated for i=2,...,n.

$t_{\pi_i} \equiv t_{[i]}$ the time required to investigate the i targets $\pi_1, \pi_2, \ldots, \pi_i$, in that order.

$I_j(t_i)$ = the time required to investigate target j given that investigation of target i is complete at time t_i.

Let $t_{[0]} = 0$. Then

$$t_{[i]} = \sum_{j=0}^{j=i-1} I_{[j+1]}(t_{[j]}) .$$

If $t_{[i-1]} < a_{[i]}$, then $I_{[i]}(t_{[i-1]})$ includes a delay in the commencement of investigation of target i due to its nonavailability at time $t_{[i-1]}$.

Until generalizations are presented in Section VII, th objective considered exclusively is that of minimizing the number of targets to escape uninvestigated, which is equivalent to maximizing the number of elements in the solution permutation π. The latter form is used and described by defining $|\pi|$ as the number of elements in π, and the problem thus is to determine π which will

$$\text{maximize} \quad |\pi|$$

s.t.

$$t_{\pi_i} \leq e_{\pi_i}, \quad \pi_i \in \pi \qquad (2\text{-}1)$$

Constraints (2-1) require that the investigation of each target contained in π be completed prior to its escape.

In an n target problem there are $\sum_{m=1}^{n-1} \binom{n-1}{m} m!$ possible solutions, the investigator always being designated as target number one. The set of feasible solutions consists of all π for which constraints (2-1) are satisfied. Note that an n target problem contains $\sum_{m=1}^{n-1} m\binom{n-1}{m} m!$ constraints.

C. DISCUSSION

The fundamental nature of Investigation problems is that of optimization over a discrete set, identifying them to be problems in discrete mathematics. Unfortunately, there does not exist a unified theory of discrete mathematical optimization [Ref. 37]. Investigation problems are thus defined and treated as a special case.

The meaning of "investigation" is left undefined in general and is made a requirement of the particular problem formulation, but is intended to be an action carried out by the investigator, with respect to a target, which is desirable for some reason.

When applicable, investigation times include the time of any travel required in the conduct of the investigation. It can thus be seen that these times may be sequence dependent functions of the time investigation is initiated. In a job shop context, this corresponds to combined set-up and processing times which are both sequence dependent and functions of start times. The existence of just sequence dependence very seriously complicates the problem as evidenced by the great interest in and difficulty associated with obtaining solutions for the traveling salesman problem. A great deal of theory has been and continues to be developed in support of the search for an efficient, direct solution technique for the traveling salesman problem, and one might suspect that these ideas would be useful in solving investigation problems. Although existing traveling salesman theorems [Ref. 4] reveal characteristics of optimal solutions for that problem, they do not even apply to solutions for investigation problems due to the movement of targets from their initial locations and the criteria of optimization. This is illustrated through consideration of several theorems presented by Barachet and Flood [Refs. 3 and 17] and discussed by Bellmore and Nemhauser [Ref. 4].

Theorem 2 of Ref. 4 states that under the Euclidean distance measure, there exists an optimal traveling salesman tour which will not cross itself. The Euclidean measure is clearly applicable in Investigation problems, yet this theorem is not, as illustrated by the following example

shown in Figure 2.2. Let the problem situation be that described for the example of Figure 2.1 with all targets moving directly toward the border at the same speed. Let the investigator's maximum speed be the same as the target's. The dotted path is clearly optimal since it is the only feasible path containing all targets, yet the path crosses itself.

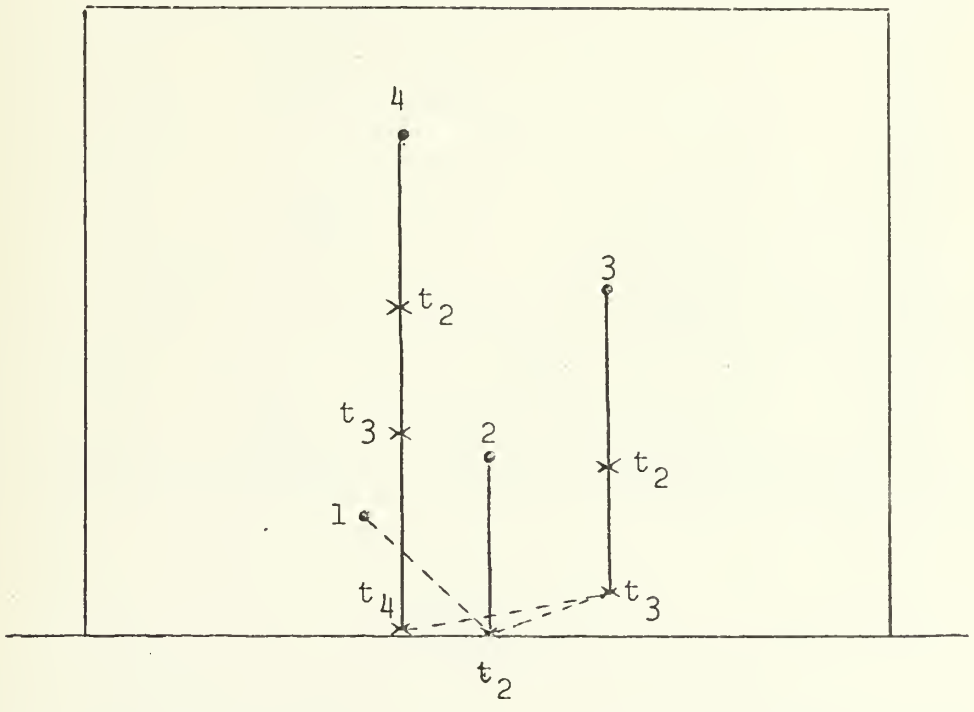


Figure 2.2

Theorem 3 of Ref. [4] states that if G is the convex hull of targets in two-dimensional Euclidean space, then there exists an optimal tour in which the relative order of the points on the boundary of the convex hull is preserved.

Using the same example situation with initial positions as shown in Figure 2.3, the dotted path again is the only feasible path containing all targets, yet the relative order of points on the convex hull is not preserved.

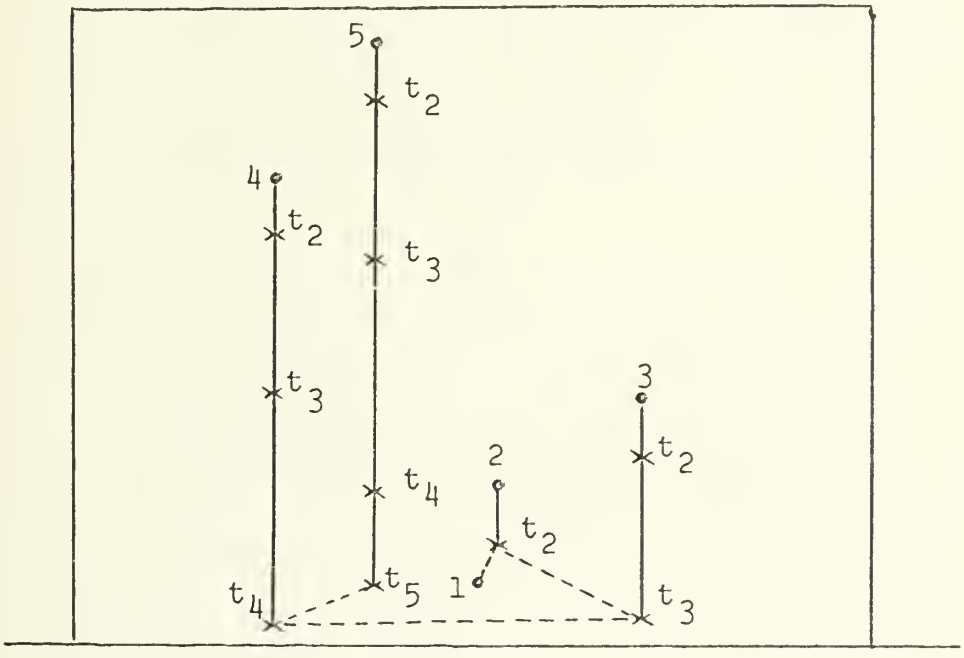


Figure 2.3

The complete constraint set for a problem is represented implicitly by constraints (2-1) because the $\sum_{m=1}^{n-1} m\binom{n-1}{m}m!$ constraints are too numerous to be explicitly stated. For example, in a problem with the investigator and only ten targets, this number is 88,776,910.

The integer nature of the variables suggests that integer programming is an appropriate approach to optimization, and,

with sufficient ingenuity, most problems treated to date have such a formulation. Unfortunately, almost all of these formulations are of negligible computational interest due to the enormous number of constraints. An exception will be presented in Section V.

One of the objectives of this research is to obtain optimal solutions to investigation problems for use in analysis of problem structure and also in the development of practical heuristic solution methods. It was thus necessary to develop an algorithm for obtaining optimal solutions.

The only problem other than the traveling salesman problem bearing a resemblance to the investigation problem which is treated to any extent in the literature is the job-shop scheduling problem with due-dates and sequence dependent set-up times. Even then the resemblance is restricted to solution and constraint sets in that the criteria of optimization, "minimize the number of late jobs," remains essentially untreated. The problem considered by Moore and others, [Refs. 1 , 13, 30, and 32] uses this criteria, but processing times are sequence independent. It is argued that problems with this structure are best treated through application of combinatorial programming techniques [Ref. 16]. It was thus decided to develop a branch and bound algorithm for obtaining the needed optimal solutions to the general problem presented in Section II,B.

D. SUMMARY OF PROBLEM TREATMENT

In Sections III through VI the problem of Section II,B is considered as follows.

Section III treats investigation problems in which a known ordering is imposed on the set of all feasible solutions. This class of problems is shown to be efficiently solved through a dynamic programming formulation. It is shown that even more efficient methods are possible through the development of a simple matrix algorithm for a certain subproblem.

In Section IV, the general problem of Section II,B is analyzed. Optimal solutions to randomly generated sample problems are used to illustrate how heuristic solution methods can be developed and evaluated. The algorithm used to obtain the optimal solutions is presented and illustrated by application to a particular problem formulation. The most general problem which can be expected to be solved using the algorithm is identified in terms of target and investigator motion characteristics.

In Section V the notion of uncertainty is introduced through consideration of the problem in which sequence independent investigation times are known only in probability distributions. The sequence independence feature allows an integer programming formulation in which the number of constraints is not prohibitive. It is illustrated that when such formulations are possible, the chance constraint

technique of Charnes and Cooper may be applicable. A very efficient algorithm which is proved to produce optimal solutions for this formulation is presented.

In Section VI problems in which targets are not available at problem time zero are analyzed. Theorems are developed and used in the construction of direct solution methods for several types of problems, and in the development of significant accelerations for combinatorial approaches.

Section VII contains a discussion of several generalizations to the problem, not within the scope of this research, which appear to be promising prospects for future research.

III. SPECIFIED ORDERING

A. PRELIMINARY REMARKS

A group of problems which can be efficiently solved includes those for which all targets are available at time zero and a known ordering is imposed on the set of all solutions. This corresponds to having a solution space containing only the $\sum_{i=1}^{i=n-1} \binom{n-1}{i}$ subpermutations of a single permutation π, i.e., all subsets of elements $\{\pi_2,\ldots,\pi_n\}$ ordered as in π. The number of constraints on such a problem is thus reduced to $\sum_{i=1}^{i=n-1} \binom{n-1}{i}(n-i-1)$.

The known ordering, corresponding to the order in which indices appear in π, can arise in a variety of ways. For example, in the case where investigator's motion is restricted to the border, either investigations are performed in the order in which targets reach the border, (escape time order), or not at all. Jobshop scheduling problems with scheduling disciplines imposed which are formulated as investigation problems also have this characteristic in that all optimal schedules must conform to the order specified by the discipline. As an example, the first-come, first-served discipline corresponds to investigating targets in the order in which they arrive in the region. (Note also that the escape time order corresponds to the due-date order.)

If such an ordering exists, combinatorial complications are greatly reduced. This is seen by recognizing that the

decision problem reduces to excluding the minimum number of indices from π in order to produce feasibility, and that the search for the optimum is guided by the known ordering.

Many potential applications of the results obtained for this problem are known, including: the defense of a coastline from infiltrators as discussed in Section I, point and area defense problems, and submarine barrier problems.

One methodology which can be used to solve such problems (see Ref. 2) is dynamic programming, as illustrated in the following section. Even more efficient methods are possible for special cases, as shown in Section III,C.

B. DYNAMIC PROGRAMMING FORMULATION AND SOLUTION OF PROBLEMS HAVING A SPECIFIED ORDERING

To clarify presentation and aid in understanding, the dynamic programming formulation is described in terms of a simplified version of the problem in which all targets have the same motion directly toward the border at the same speed, and the investigator is required to remain on the border, (thus specifying an escape time order). Investigations are considered complete when the position of the target is reached. This version is easily solved using a graphical technique guided by the dynamic programming principle of optimality. Afterwards, generalizations are presented which consider problems with more than one investigator, and specification of arbitrary orderings on solutions is allowed. Further generalizations include arbitrary courses and speeds

for the targets, alternate meanings of investigation, and different objectives.

1. Dynamic Programming Formulation of Problems With One Investigator on the Border

In the problem being considered there are n targets located in a rectangular region with initial locations given as (x_i, y_i) for i = 1,...,n. See Figure 3.1. Interval [0,R] of the x-axis is the border of the region and all targets are proceeding directly toward the border at speed s_2. Initially the investigator is required to remain on the border. His objective is to investigate as many targets as possible as they reach the border. The investigator has maximum speed s_1 and can reverse direction instantaneously.

This problem is formulated as an n stage dynamic programming problem [Ref. 33], as follows. Order and index all targets according to decreasing distance from the border. Assign all targets with equal distances the same index. Let stage j correspond to the time target j+1 arrives at the border. Stage n is thus the beginning of the problem. The state variable X_j represents the position of the investigator on the border at stage j, and $f_j(X_j)$ is the maximum number of targets which can be investigated prior to escape from among targets 1,...,j, given state X_j. The decision d_j at stage j is to move along the border from X_j to X_{j-1}. The return at each stage, $r_j(X_j,d_j)$, equals one if target j is investigated and zero otherwise.

Let e_j be the time target j reaches the border and define $\bar{t}_j = e_j - e_{j+1}$. The recursive equations are

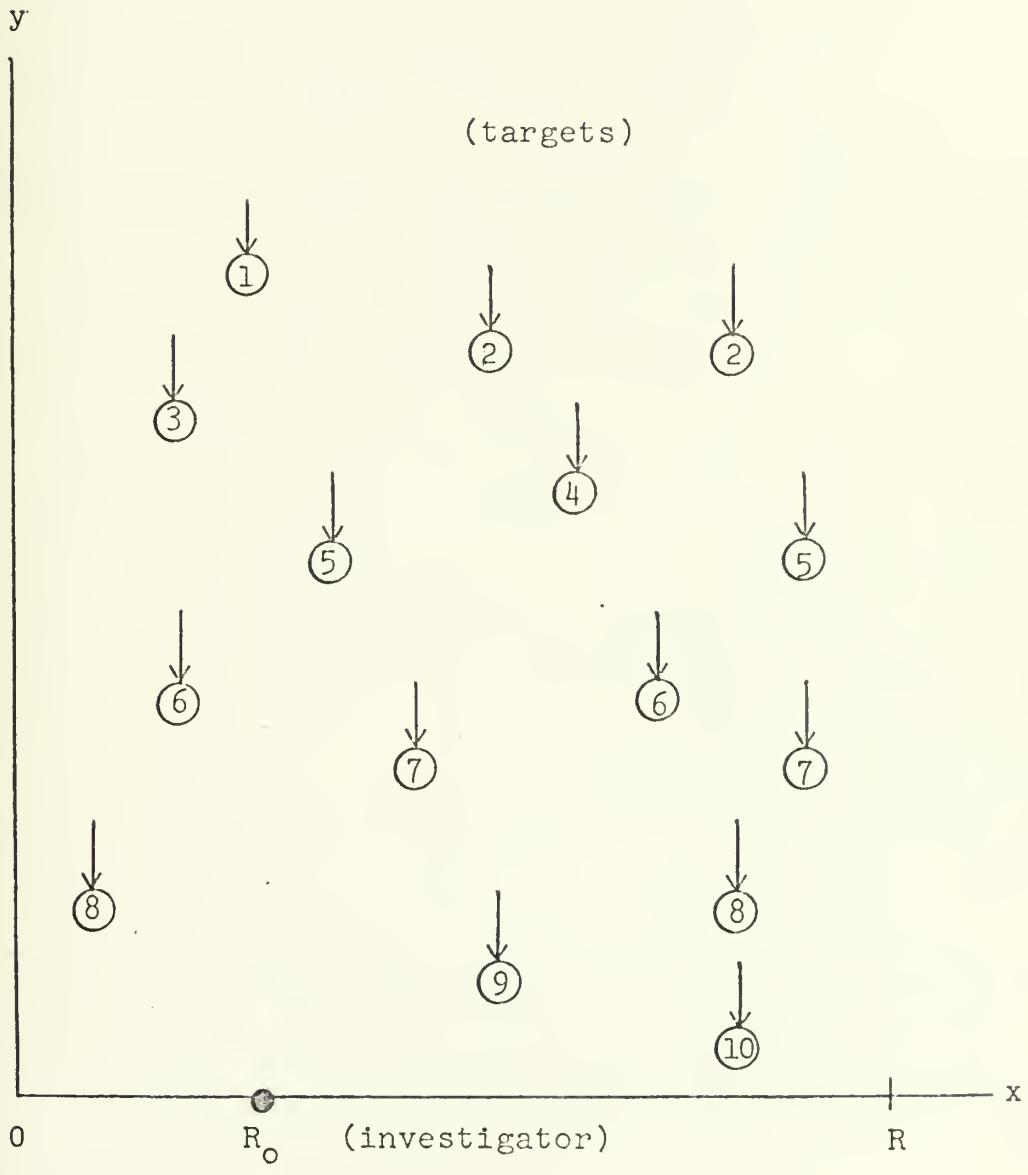


Figure 3.1. An investigation problem

$$f_1(X_1) = \max_{d_1} r_1(X_1,d_1)$$

and

$$f_j(X_j) = \max_{d_j}\{r_j(X_j,d_j) + f_{j-1}(X_{j-1})\}$$

where

$$X_{j-1} = X_j + d_j$$

and

$$-s_1\bar{t}_j \leq d_j \leq s_1\bar{t}_j$$

and

$$-X_j \leq d_j \leq R - X_j.$$

The first constraint on d_j limits the maximum distance which the investigator can move during time $\bar{t}_j$ and the second requires that the investigator remain in the interval [0,R].

Note that the return function is

$$r_j(X_j,d_j) = \begin{cases} 1, & X_j + d_j = x_j \\ 0, & X_j + d_j \neq x_j \end{cases}$$

Consider the situation where k targets arrive at the border simultaneously. At most one target can be investigated, hence $r_j = 1$ if $X_j + d_j = x_j$ for any such target. Note that if this occurs that the problem is reduced to an n-k+1 stage problem since all k targets are assigned the same index.

2. Graphical Solution for the Problem With One Investigator on the Border

The recursive equations can be solved in the usual way to obtain an optimal solution, however, for this problem, a simple graphical method is also possible.

In the graphical method, decisions are simplified by noting that at stage j, given any state X_j, the determination of whether target j can be investigated can be resolved by drawing a cone with vertex at the position of target j and noting whether X_j is contained inside the cone. See Figure 3.2. The shape of the cone is determined by $\bar{t}_j$ and s_1. Even if target j can be investigated given state X_j, it may not be optimal to do so since moving to x_j may put the investigator in a bad position with respect to targets 1,2,...,j-1. This condition is reflected in $f_{j-1}(X_{j-1})$ which is recorded for each possible point X_{j-1} on the transposed border passing through target j.

The graphical solution is carried out by recursively drawing the border through each target's initial position and recording for each stage 2,...,n the quantity $f_j(X_j)$ which specifies the maximum number of investigations possible.

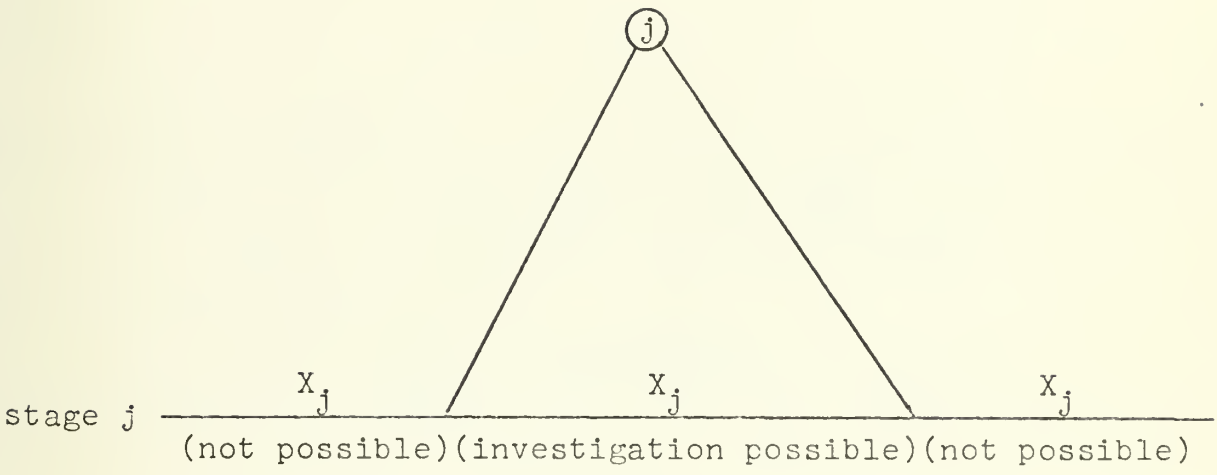


Figure 3.2 Cone for a single target.

The method is illustrated with an example. For the problem pictured in Figure 3.1, let $s_1 = 2$ and $s_2 = 1$. The solution is shown in Figure 3.3. The numbers on the horizontal lines give the values of $f_j(X_j)$. The optimal solution shows that the investigator can at best investigate six targets.

3. Multiple Investigators on the Border

The dynamic programming solution method presented for the problem with a single investigator is computationally simple. The formulation given is also applicable to the problem of scheduling m investigators against n targets to maximize the number of targets investigated. In this

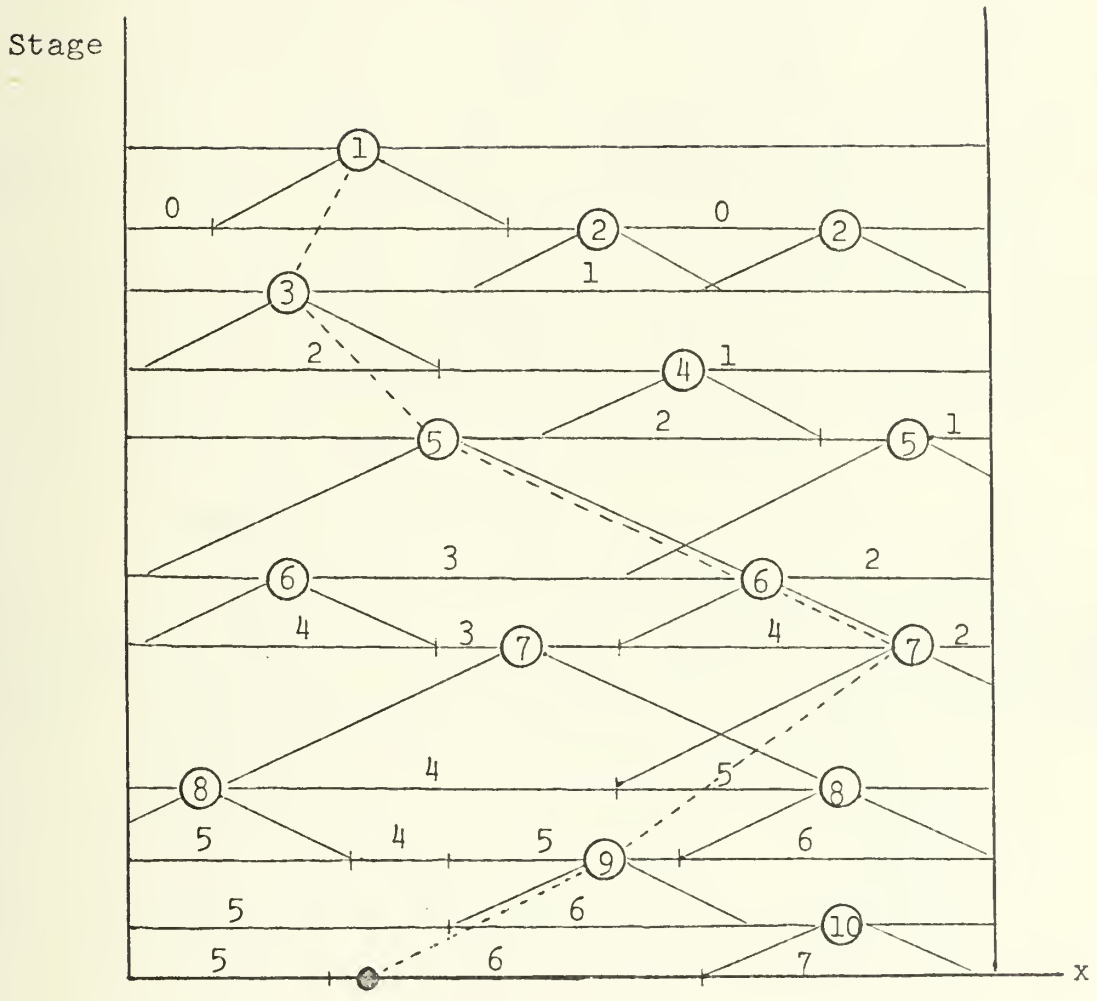


Figure 3.3. Graphical solution to example problem. (An optimal sequence is indicated by the dotted line.)

section, the problem considered is one in which the investigators all stay on the border and are permitted to cross one another. Their maximum speeds may differ. To handle this generalization, it is only necessary to interpret the state variable as an m-vector whose components give the positions of each investigator at the time a target reaches the border. This is easily visualized, and easily computed for the case m = 2. The state space simply becomes a portion of the plane. Regions in the space are recursively labeled to indicate the maximum number of remaining targets which can be investigated from that point in the space. The method is illustrated with an example.

4. Example and Solution

In this section a problem having m = 2 and n = 5 targets is presented and solved. Table 3.1 gives the target data. Investigator one has speed of one unit per unit time

j	x_j	e_j	$\bar{t}_j$
1	0	8	1
2	3	7	1
3	6	6	3
4	10	3	1
5	5	2	2

Table 3.1. Target data
for sample problem with m = 2.

and investigator 2 has a speed of 2. Figures 3.4 through 3.9 show the original problem and the functions $f_j(\overline{X}_j)$, $j = 1,\ldots,5$ where X_{ij}, $i=1,2$ is the position of investigator i at stage j.

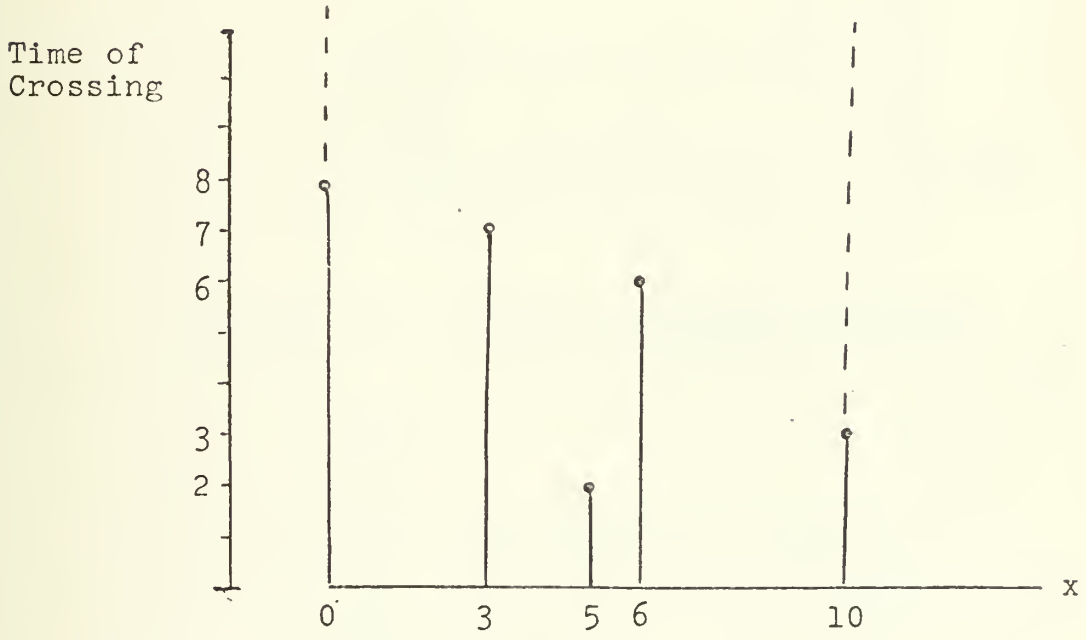


Figure 3.4. Problem data for example with m = 2

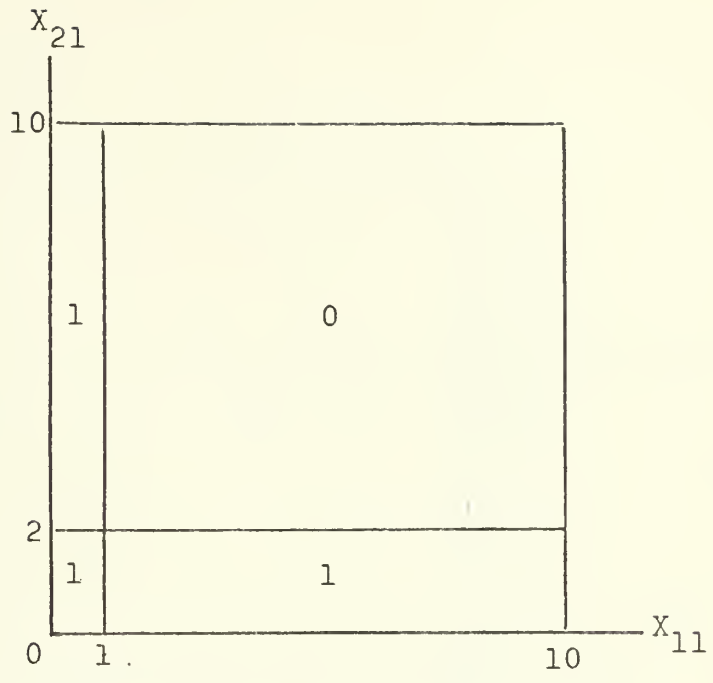


Figure 3.5. $f_1(\overline{X}_1)$ for example problem

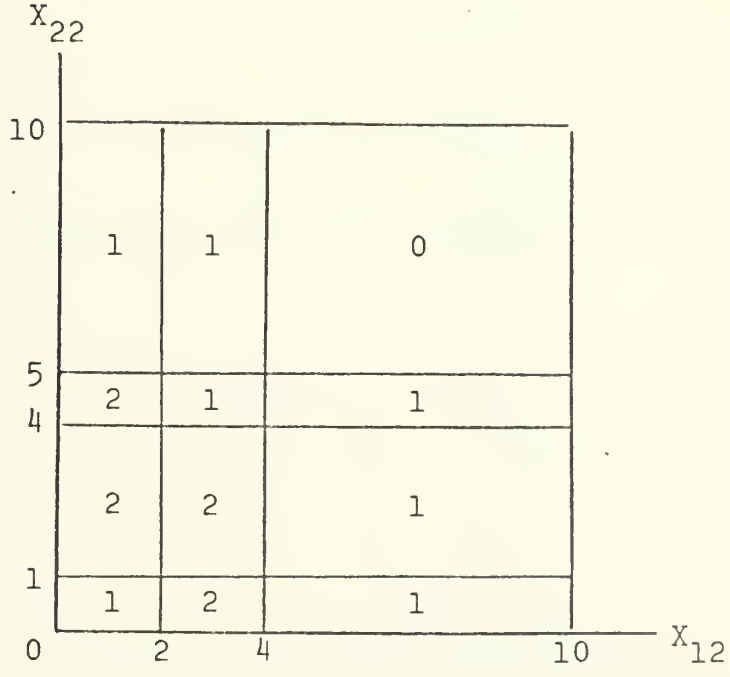


Figure 3.6. $f_2(\overline{X}_2)$ for example problem

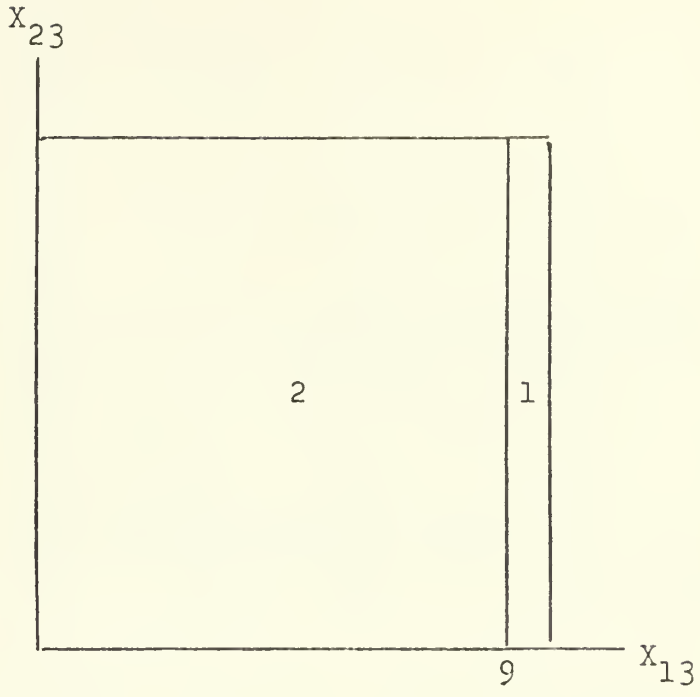


Figure 3.7. $f_3(\overline{X}_3)$ for example problem

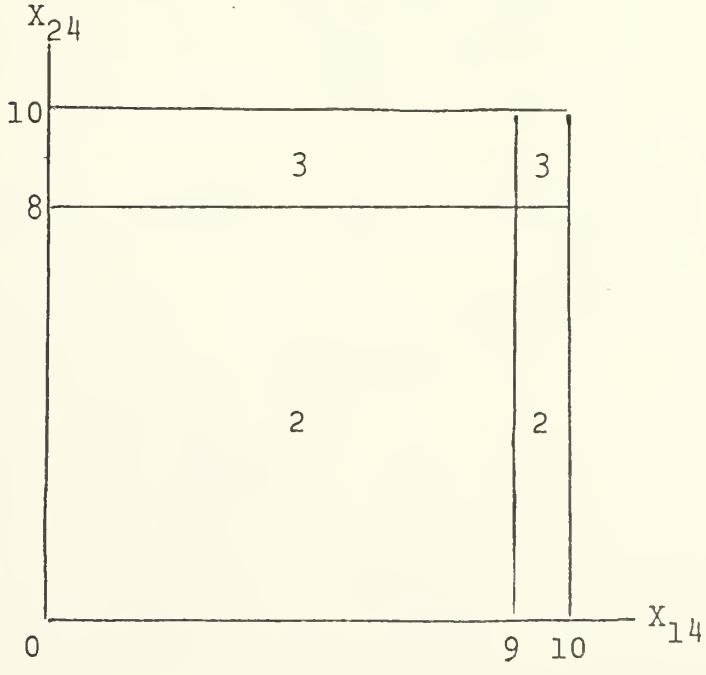


Figure 3.8. $f_4(\overline{X}_4)$ for example problem

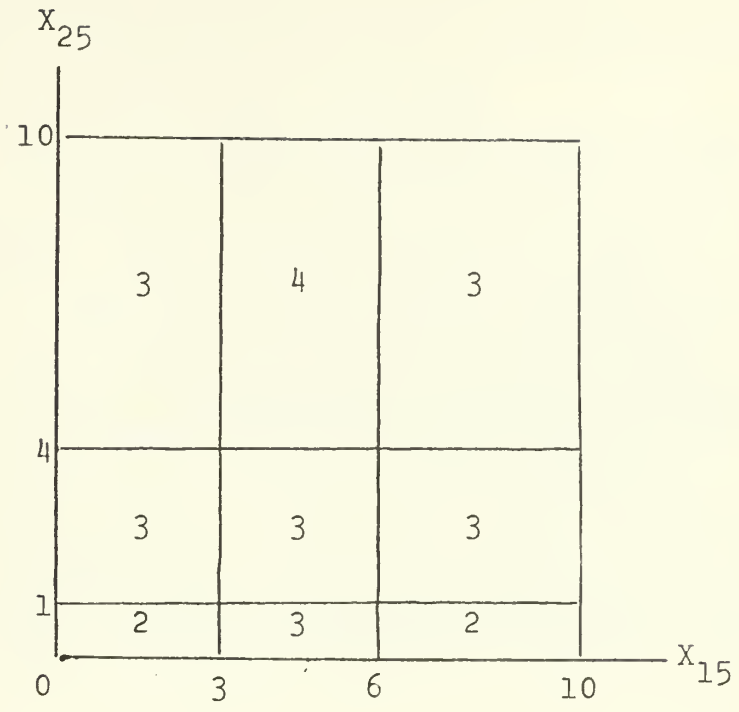


Figure 3.9. $f_5(\overline{X}_5)$ for example problem

Note from function $f_5(\overline{X}_5)$ that the two investigators can at best investigate four targets and this can occur only if investigator one has a starting position between three and six and investigator two has a starting position to the right of four. In the worst case, corresponding to the regions labeled with a two, the investigators will be able to investigate only two targets.

5. Arbitrary Orderings

In order to consider arbitrary orderings on the solution set, it is necessary to relax the requirement that the investigator(s) remain on the border. To solve such problems using dynamic programming, the state variable $\overline{X}$

must have m+1 components where m is the dimension of the region. The first m components give the position of the investigator and the other gives the time.

Stage j is again identified by the $j+1^{st}$ target reaching the border. The decision is which target to investigate next. The solution be dynamic programming would require imposing an m-dimensional grid over the region R and performing the standard dynamic programming calculations. For problems with more than one investigator, the dimension of $\bar{X}$ increases by m for each additional investigator. Consequently, dynamic programming becomes impractical very rapidly.

6. Further Generalizations

When regarding the state variable $\bar{X}$ as a vector, other slightly different interpretations of the problem are possible. Returning to the single investigator problem, it is possible to permit the border to be m dimensional as would be the case when the investigator is guarding a portion of a plane. See Figure 3.10. In this case, the components of $\bar{X}$ are simply the position of the investigator on the plane.

Likewise, some components of $\bar{X}$ can be interpreted to be descriptions of the physical condition of the investigator. For example, it is possible to permit the investigation of a target to change the maximum speed at which the investigator can travel. This could be used for the case in which investigation of some target is a dangerous

operation and results in damage to the investigator. In fact, the occurrence or non-occurrence of damage could be permitted to be a random event.

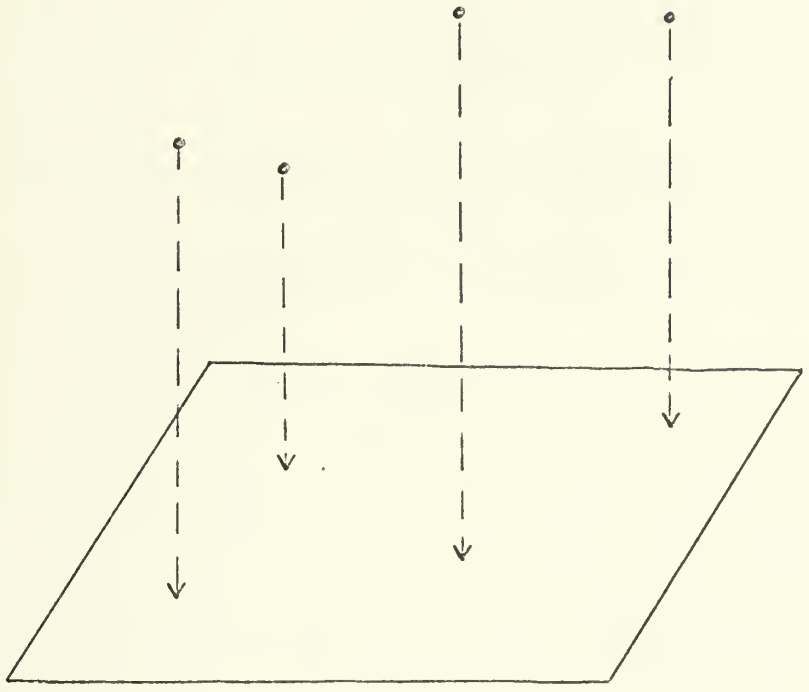


Figure 3.10 A two dimensional boundary defense problem.

For the problem in which the investigator must remain on the border the solution procedure can easily be modified to include several other generalizations in addition to interpreting the state variable as a vector. For example, each target can move with a different speed and heading. The speed will affect the width of the cone and the heading will affect its projection on the boundary. In fact, it doesn't matter where the target begins or what path it follows to reach the boundary. It can move in any manner at all as

long as its point and time of crossing are known. For such cases where targets are approaching the border at different speeds, it is necessary to order the stages (number the targets) so that they arrive in the order n, n-1,...,1.

Also note that a value or priority V_j can be assigned to each target to reflect the importance of investigating it. This would be done by letting $r_j(X_j,d_j) = V_j$ if interception is made and zero otherwise. The problem would remain one of maximizing the V_j's summed over those targets which were intercepted.

In another generalization it is possible to assume investigation is complete when the investigator is within a distance d of the target when he reaches the border. It is also possible to solve the problem for the case in which each target remains on the border for a finite time before penetrating. The case in which a finite processing time is required for each target is also handled easily. Likewise it is easy to deal with cases where the investigators are given different maximum speeds, perhaps zero, in moving left or right.

C. A MATRIX ALGORITHM FOR PROBLEMS WITH A SPECIFIED ORDERING

Under certain conditions it is possible to develop even more efficient algorithms for obtaining optimal solutions. Such a problem is one in which targets must be investigated in a specified order, the investigator is allowed off the border, and target motion is directly toward the border at

the same speed, (see Ref. 41). In this case, investigation times are sequence dependent but independent of start time, i.e., $I_j(t_i) = I_{ij}$, for $i = 1,\ldots,n$, $j = 2,\ldots,n$.

1. The Algorithm

Let $\pi = (\pi_1,\pi_2, \ldots ,\pi_n)$ specify an ordering to which all optimal solutions must conform. If π is feasible, i.e., if

$$t_{\pi_i} \leq e_{\pi_i} \quad \text{for } i = 2,3,\ldots,n,$$

this sequence is optimal. If not, one or more target indices must be excluded from π to produce a subpermutation which is feasible. Suppose π_j is the first index in π such that $t_{\pi_j} > e_{\pi_j}$, then from among indices $\pi_1,\ldots,\pi_j$, that index is excluded from π which minimizes the time required to investigate the remaining targets. This rule is repeated each time an exclusion is required.

The steps in the algorithm are as follows.

1) Index targets according to the order specified in π and form the upper triangle of the n by (n-1) matrix (I_{ij}) of elements I_{ij} for $i < j$.
2) Augment the matrix (I_{ij}) with an additional row which contains the escape times for the targets.
3) Compute the completion time t_i, $i = 2,\ldots,k$ where

$$t_1 = 0, \quad t_i = t_{i-1} + I_{i-1,i}.$$

For the first k such that $t_k > e_k$, go to step 4. If $t_i \leq e_i$, i = 2,...,n, then the minimum number of targets to escape is zero.

4) For each column 2,...,k compute the savings S_i where

$$S_i = t_{i+1} - (t_{i-1} + I_{i-1,i+1}).$$

5) From the matrix, delete row and column h such that

$$S_h = \max \{S_i, \quad i = 2,...,k\} .$$

Go to step 3 and recompute $t_{[i]}$'s starting with target h+1. (Note that after the matrix is reduced, the subscripts refer to relative positions of columns in the matrix rather than target numbers in the computational formulas for t_i and S_i.)

The set of columns in the final matrix specifies the largest set of targets which can be investigated before their escape.

2. Example

The algorithm will be illustrated with the following example. Let the augmented matrix of step 2 be that shown in Figure 3.11.

	2	3	4	5	6	7	8	9
1	21	13	35	14	29	35	24	32
2	21(20)	12	15	26	11	16	18	20
3	39	33(23)	26	15	16	22	14	20
4		40	59(57)	46	11	11	30	24
5			48	105	30	35	13	24
6				50		5	21	13
7					70		29	15
8						80		11
9							81	
10	39	40	48	50	70	80	81	100

Figure 3.11 The Augmented Matrix

The elements from row 10 have been moved up under the main diagonal for convenience. Step 3 has been carried out until $t_4 > e_4$. t_i's are shown in the main diagonal spaces. Step 4 has been completed with S_i's shown in the parenthesis in the main diagonal spaces. The quantity S_4 is max $\{S_i, i = 2,3,4\}$, so row and column 4 are to be deleted prior to returning to step 3.

The reduced matrix is shown in Figure 3.12 and again steps 3 and 4 have been completed showing max $\{S_i, i = 2,3,5,6\}$ to be S_5. Row and column 5 are deleted and step 3 is returned to again. The resulting reduced matrix is shown in Figure 3.13.

	2	3	5	6	7	8	9
1	21	13	14	29	35	24	32
2	21(20)	12	26	11	16	18	20
3	39	33(1)	15	16	22	14	20
5		40	48(29)	30	35	13	24
6			50	78(0)	5	21	13
7				70	83	29	15
8					80		11
9						81	
10							100

Figure 3.12 The First Reduced Matrix

	2	3	6	7	8	9
1	21	13	29	35	24	32
2	21(20)	12	11	16	18	20
3	39	33(17)	16	22	14	20
6		40	49(-1)	5	21	13
7			70	54(13)	29	15
8				80	83(25)	11
9					81	94
10						100

Figure 3.13 The Second Reduced Matrix

Steps 3 and 4 show now that S_8 is maximum, so proceeding as before, row and column 8 are deleted.

	2	3	6	7	9
1	21	13	29	35	32
2	21	12	11	16	20
3	39	33	16	22	20
6		40	49	5	13
7			70	54	15
9				80	69
10					100

Figure 3.14 The Solution Matrix

It can be seen in Figure 3.14, after deleting row and column 8 and carrying out step 3 that all remaining $t_i \leq e_i$ and the optimal permutation is (1,2,3,6,7,9). This solution also has the property that for all solutions in which only three targets are not investigated, this permutation completes investigation of those investigated in minimum time.

3. Proof of Optimality

The algorithm is now proved to produce an optimal solution. The proof is similar to those used in references 13 and 32 for the nearly equivalent jobshop problem in which processing and set-up times are sequence independent. Targets are separated into two disjoint sets, E and L, corresponding to those targets which are investigated early and those which are not according to the current subpermutation being considered. Define

E_k = those targets out of the first k which have been retained in the investigation schedule.

L_k = those targets out of the first k which have been excluded from the investigation schedule.

$|E_k|$ = the number of targets in set E_k. (Similarly for $|L_k|$).

Given that out of the first k targets in the sequence $|L_k|$ must be excluded, then it is optimal to place those targets in L_k which allow the $|E_k|$ targets in E_k to be completed in minimum time since this will allow the remaining n-k targets the greatest opportunity to be investigated before their escape times.

The optimality of this procedure can be seen by supposing that for some current sequence with $|L_k| = j$, $2 \leq k < n$, the next target in the sequence has $t_{[k+1]} < e_{[k+1]}$ In this case $|L_{k+1}| = j$ and $t_{[k+1]}$ is minimal. On the other hand, if $t_{[k+1]} > e_{[k+1]}$ then $|L_{k+1}| = j+1$. The target to be placed in L_{k+1} is target r where r is in $E_k \cup \{k+1\}$ and $S_r = \max_i \{S_i : T_i \text{ in } E_k \cup \{k+1\}\}$. Then $t_{[k+1]} - S_r \leq t_{[k+1]} - S_i$, i in $E_k \cup \{k+1\}$ Hence the adjusted time to complete investigation of target k, given that one more target has been placed in L_k, is minimized by selecting r for placement in L_k. Optimality is thus proved.

4. Discussion

It is instructive to note that this algorithm reduces to Moore's well known algorithm [Ref. 32], if investigation times are sequence independent. If $I_{1j} = I_{2j} = \ldots = I_{nj}$ for all j, i.e., if investigation times for all targets are independent of sequence, then the elements of the j^{th} column of the problem matrix are all equal. The algorithm presented will in this case choose that target out of the first k with the largest investigation time and place it in L. This rule is exactly the one used by Moore while obtaining optimal solutions to the equivalent job-shop problem with sequence independent set-up times.

This Section treats only problems whose solution space consists of subpermutations of a single permutation. This characteristic allows efficient solution methods to be developed. When the sole optimal solution may be a subpermutation of several permutations, obtaining solutions becomes much more difficult. Such problems are the subject of the following Section.

IV. EVALUATION OF HEURISTIC SOLUTION METHODS

A. INTRODUCTION

This Section deals with the general combinatorial optimization problem presented in Section II in which there are no external limitations on sequence. The solution space for this problem is a discrete set of all subpermutations of the (n-1)! permutations of the integers 2,3,...,n. When more than one permutation may contain the optimal subpermutation, it is necessary to explicitly or implicitly evaluate all such permutations in any search for the optimum. The degree of difficulty associated with a problem is directly related to the number of permutations which may contain the optimum.

Experience in solving the general problem shows that the expected number of computations required tends to rise exponentially with n, the number of targets. Optimal solutions for even small problems, (10 targets), cannot be expected to be obtained without the assistance of a computer. The actual environment in which the problem exists may make obtaining optimal solutions impossible in most cases and impractical in the rest because computers usually are not available. Even when they are, the time allotted for decision-making may make uncertainty in solution time unacceptable. In such cases the decision-maker has no alternative but to devise and use heuristic solution methods.

If it is necessary to use a heuristic, it is clearly desirable to use one which is known to be good in some sense. Determination of this fact requires the existence of a measure of effectiveness or efficiency for such rules. It is felt that the only true measure of an heuristic solution method is through comparison of solutions obtained with optimal solutions. Other relative measures are clearly possible, such as comparative testing. These measures illustrate dominance of one rule over another, but reveal little or nothing with respect to the optimum. The clear advantage in comparing heuristics to the optimum is that the decision maker can stop devising and testing new rules when one is found which produces solutions sufficiently close to the optimum. Comparative testing does not offer this choice.

In this Section, a format for evaluating heuristics is presented through consideration of one problem formulation. Three rules are devised and evaluated through comparison with optimal solutions obtained for two sets of randomly generated sample problems. Next, the branch and bound method used to obtain the optimal solutions is described. Generalizations of the method are presented, describing the most complex target and investigator motion which can be expected to be treated. A specific algorithm is then presented, in flow diagram form, which can be used to solve a large class of investigation problems. The solution for one sample problem is shown, followed by a discussion of computational experience with the algorithm.

B. EVALUATION OF THREE HEURISTIC SOLUTION METHODS

A problem description is given, followed by development and evaluation of three easy-to-use heuristic solution methods [Ref. 42].

Consider the problem where targets are located in a planar region and all are moving directly toward a prescribed border. The single investigator moves with a speed greater than that of any target and is free to change course instantaneously. Naturally, he always moves at maximum speed. To investigate a target only requires that the position of the investigator and the target coincide. Investigation times correspond to travel times between pairs of targets which are sequence dependent functions of the time investigation is initiated.

For this problem, the three heuristics listed below seem worthy of consideration. Each is an easily implemented rule which, upon repeated application, selects a feasible solution which intuition suggests may be good with respect to the optimum.

1. STP - Shortest Time Path. This rule always directs the investigator to move next to that target which is closest in time to the investigator's current position providing that the investigator can reach that target before it reaches the border.
2. SDP - Shortest Distance Path. This rule directs the investigator to move next to the target closest in distance to the investigator's current position

providing that the target can be reached before it crosses the border.

3. CTB - Closest To Border. Using this rule the investigator procedes next to the target closest to the border providing it can be reached before it crosses the border.

The solutions generated by these rules are compared to the optimal solutions by considering forty problems each of which has twenty targets whose initial positions were generated randomly from a uniform distribution. The object nearest the border is considered to be the investigator. The other 19 are the targets. The data and solutions for all problems are contained in Ref. 42.

The region considered is 10 units wide and 10 units deep. In the first set of 20 problems the investigator has a speed of 25 units per unit time, and all targets have a speed of 15 units per unit time. In the second set of problems the targets have speeds drawn from a uniform distribution between 10 and 20 units per unit time.

A square region is used because this appears to present the most difficult problem for a fixed number of objects. A wide, shallow region will allow the investigator to capture only those few targets relatively near his position, most of the others being out of reach even if he goes toward them immediately. A narrow, deep region will present little

difficulty since the investigator will normally be able to capture a large fraction of the targets with any reasonable sequence.

1. Same Target Speeds

The results for this set of problems is summarized in Tables 4.1a and 4.1b, showing for each problem the number of targets investigated in the optimal solution and with each of the three rules. For each problem the ratio, called the effectiveness ratio, of the number of targets investigated using the rule to the number investigated optimally is computed and the average of these ratios is shown in Table 4.1b.

Problem	OPT	RULE STP	SDP	CTB	Problem	OPT	RULE STP	SDP	CTB
1	11	10	10	8	11	11	10	10	8
2	11	9	9	11	12	9	7	8	9
3	11	8	7	6	13	12	10	6	12
4	11	10	11	6	14	10	10	10	9
5	13	12	13	12	15	12	12	12	12
6	12	10	12	10	16	14	10	13	12
7	11	9	9	11	17	10	8	10	8
8	11	10	10	9	18	9	8	9	8
9	9	9	9	9	19	9	7	9	9
10	9	5	9	6	20	10	9	10	10

Table 4.1a. Summary of Results for Problem Set One

RULE	AVERAGE OF EFFECTIVENESS RATIOS
STP	.85
SDP	.92
CTB	.87

Table 4.1b. Effectiveness Ratios for Problem Set One

It can be seen from Table 4.1b that for these twenty problems the best of the three rules is SDP. The STP and CTB rules appear to be less effective. The deficiency of the STP rule is that it has a tendency to move the investigator outward from the border toward incoming targets, foregoing nearby targets which might later be impossible to capture since they would have to be "run down" from behind. The CTB rule suffers from the deficiency that it causes the investigator to make more movements parallel to the boundary than might otherwise be required. Time is thus wasted in back and forth motion.

Each of the rules is nearsighted and looks only one step ahead, but in view of the results presented here it would appear that if computer equipment is not available to compute optimal solutions, the SDP rule should be applied. It is easier to apply than the STP rule since no relative motion calculations need to be made.

When viewed in job-shop context, the CTB rule represents the rule "process next the job nearest its

due-date," and the STP rule the "nearest neighbor" rule, (or process next that job with the shortest processing time). The results contained in Table 4.1b indicate that there is no substantial difference between these two rules. The SDP rule has no counterpart in the job-shop problem.

Figure 4.1 provides a pictorial representation of one of the problems just for illustration. It must be remembered that the targets are moving and the actual ground track of the investigator is not that shown in the figure.

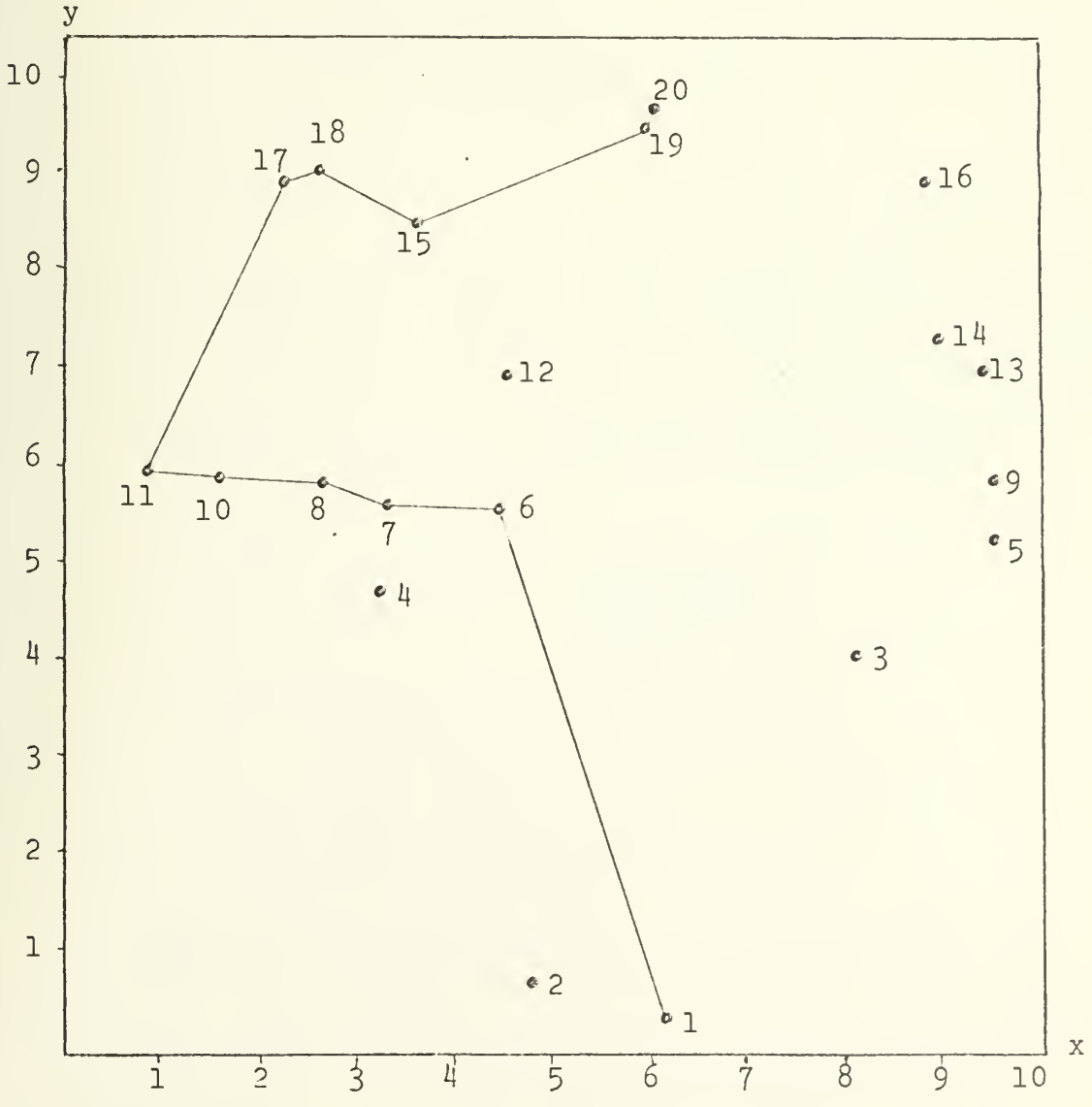


Figure 4.1 Sample Problem With Optimal Solution

2. Different Target Speeds

The results for this set of problems are summarized in Tables 4.2a and 4.2b.

Problem	RULE				Problem	RULE			
	OPT	STP	SDP	CTB		OPT	STP	SDP	CTB
1	13	10	9	11	11	12	10	11	11
2	11	8	10	11	12	12	11	11	12
3	12	8	12	5	13	10	5	8	7
4	10	10	10	9	14	12	7	11	10
5	10	10	10	9	15	11	9	9	10
6	10	8	9	9	16	11	8	9	10
7	10	10	10	9	17	10	7	9	10
8	12	9	9	11	18	11	9	11	6
9	13	11	11	11	19	11	9	10	11
10	12	10	11	11	20	16	14	15	15

Table 4.2a Summary of Results for Problem Set Two

RULE	AVERAGE OF EFFECTIVENESS RATIOS
STP	.80
SDP	.90
CTB	.87

Table 4.2b Effectiveness Ratios for Problem Set Two

The results from these 20 problems with different target speeds tend to substantiate the results given for the previous set in that the SDP rule is more effective than either the STP or CTB rules in approximating the optimum.

Using the data of Ref. 42, other heuristic solution methods can be suggested and evaluated. A multitude of heuristics have been developed in the closely related job-shop scheduling area, for example, see Refs. 6 , 19, and 20.

C. SYNOPSIS OF ALGORITHMS USED FOR OBTAINING OPTIMAL SOLUTIONS

1. Preliminary Remarks

Optimal solutions being required in the evaluation of heuristic methods, it was necessary to develop a means of obtaining these solutions. The methodology of branch and bound [Ref. 26], was used for this purpose. Possible advantages of the branch and bound method for such problems is discussed in Ref. 16, and examples of applications to similar job-shop scheduling problems are contained in Refs. 5 , 22, 28, and 31.

Investigation problems may differ from each other in many ways, but from a computational point of view, the most pertinent difference is in the description of target and investigator motion. As part of the intent of this research is to present analysis of a wide variety of investigation problems, it was necessary to develop algorithms for obtaining optimal solutions for several broad problem types. Each of

the algorithms possesses the same fundamental structure - an implicit enumeration of all feasible solutions as directed by the branch and bound technique, differences arising mainly from the type of motion involved.

The presentation in the next Section is a description of the branch and bound method as applied to investigation problems. This is followed by the statement of an algorithmic framework which can be used to develop specific algorithms through selection of branching and bounding rules. The ability to specify usable branching and bounding rules is shown to be dependent upon the type of motion involved. Finally, the most complex motion for which optimal solutions can be expected to be obtained is discussed.

2. Application of Branch and Bound to Investigation Problems

The branch and bound method consists of selectively partitioning the set of all feasible solutions and computing bounds on the objective function for each element of the partition. This process is continued until one element of the partition containing a single solution is obtained for which the associated bound is at least as good as that of any other element.

A feasible solution to an investigation problem is a permutation describing a path for the investigator to follow in which all targets contained in the path are investigated prior to escape, and investigation does not commence prior to target availability. For such problems, a branching

tree is initiated and extended such that each node at the end of a branch of the tree corresponds to a feasible solution. The set of all such nodes specifies a partition of the feasible solution space. The bounds associated with each element of the partition, and therefore with each node at the end of a branch of the tree, represent a lower bound on the number of targets which will escape if any solution contained in that element of the partition is followed.

Associated with every node of the tree is a target. The initial node, representing all solutions, corresponds to target number one, the investigator. Branching corresponds to selecting a node to branch from and specifying a target for use in extending the path specified by that branch. The branch corresponding to node one naturally is selected initially for extension. This branch is extended to the right and the left by adding two new nodes to the tree. See Figure 4.2. The node labeled k represents all solutions

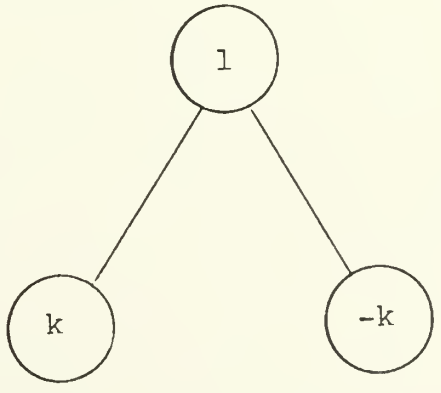


Figure 4.2. Initial Branching

in which the investigator starts from his initial position and investigates target k next. The node labeled -k represents all solutions in which the investigator does not investigate target k next.

Associated with each node at the end of a branch of the tree is a path, starting with one, and including each target number corresponding to a positively labeled node in the order in which they appear in the branch. For example, the node labeled k in Figure 4.2 corresponds to path (1,k), and the node labeled -k to path (1).

When a branch of the tree is selected for extension, the path specified by that branch is called the current path.

The bound on each ending node with a positive label applies to all feasible solutions starting with the path specified by that node. The bound on those with negative labels corresponds to all solutions starting with the path specified which do not include as their next element, all negative labels of nodes at the end of that branch. In illustration of this, consider Figure 4.3. The bound on node labeled 7 applies to all solutions starting with elements 1, 2, and 7, in that order. The bound on node labeled -2 applies to all solutions starting with element 1 and not having either 4 or 2 as the second element.

After a branch is selected for extension, it is necessary to determine which targets are eligible for use in extending the current path. This is done by determining

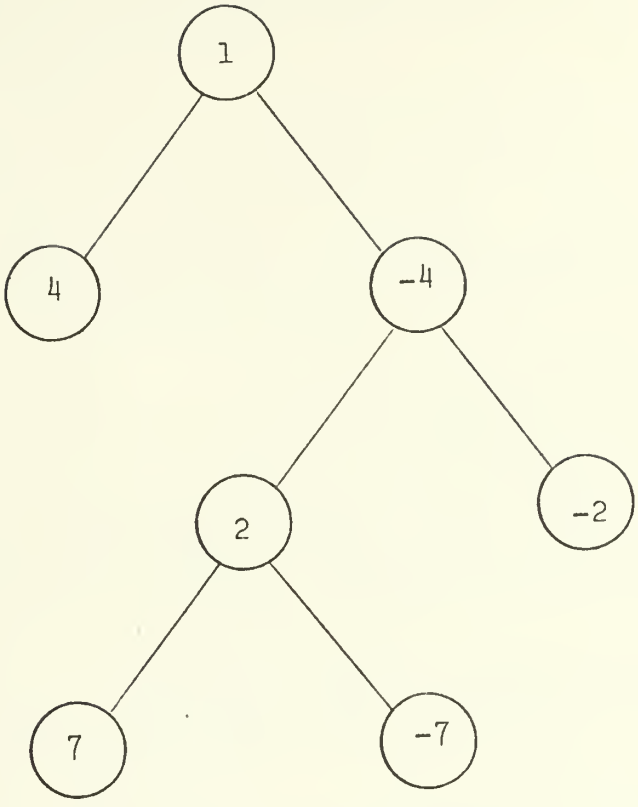


Figure 4.3 Sample Tree

all targets not in the current path and not prohibited du to negatively labeled nodes at the end of the branch, wh ch could be investigated prior to escape if investigated nex .

The order of an element of the partition reduces to one when there are no targets eligible for use in extending the branch corresponding to that element. In this case, all targets are either in the current path specified by that branch, or are known to escape as a result of following that path. When an element is located with an exact bound less than or equal to the bound on all others, the path corresponding to the sole solution contained in that element is optimal.

3. The Framework for Algorithms for Solving Investigation Problems

Specific methods of branching and bounding are ignored temporarily while the general structure of all algorithms developed to date is presented. This is followed by a discussion of difficulties associated with the selection of these rules.

Algorithm:

1. Create initial node corresponding to target number one, the investigator. Compute a bound for this node. Go to 2.
2. Select a branch to extend. This branch determines the current path. Go to 3.
3. Determine the set E of targets eligible for use in extending the current path. Put into set E indices of all targets not in the current path which could be investigated prior to escape if investigated next. Remove from E all indices corresponding to negatively labeled nodes at the end of the branch being extended. If $|E| \leq 1$, stop. If not, go to 4.
4. Select from E a target k for use in extending the current path. Extend the branch to the left and right creating two new nodes labeled k and -k. Go to 5.
5. Compute bounds for branches ending with nodes labeled k and -k. Go to 2.

Upon termination, if $|E| = 0$, the current path is optimal. If $|E| = 1$, the current path extended by the target corresponding to the element remaining in E is optimal.

4. Difficulties Associated with Selection of Branching and Bounding Rules

Branching is a two part operation, consisting of first selecting a branch to extend, and next selecting a target for use in extending that branch. It is a common practice in branch and bound to use as the first of these steps the selection of that branch whose associated bound is smallest and specifying tie breaking rules in the event they occur. This rule has been used exclusively.

The second part of the branching operation is particularly worthy of note. In this step, a target is selected for extending the current path. It is clearly desirable to select a target which will extend the current path in a manner which is optimal, or as near optimal as possible, with respect to the objective. Such a rule increases the efficiency of the algorithm by decreasing the number of branches required. If a rule were able to extend a given path through addition of a single target in a manner which is optimal or near optimal, then repeated application of only this rule should produce a feasible solution which itself is optimal or near optimal. Good rules for use in this step of the algorithm thus correspond to good heuristics as discussed in Sections IV,A and IV,B. This fact makes it

apparent that the output of the algorithm can be used to improve its efficiency. For example, an arbitrary rule can be used initially in this step, optimal solutions developed, and heuristics evaluated. The best of these heuristics can then be used as the rule at this step, increasing the efficiency of the algorithm. Additional heuristics can then be more efficiently evaluated. This learning feature was used to significantly improve the efficiency of the algorithm presented in Section IV,E.

The bound associated with each branch specifies a lower bound on the number of targets which will escape if the path specified by that branch is followed. For the case when the last node in the branch has a positive label, a loose but easily obtained bound corresponds to determining, as of the time investigation of the current path is complete, how many targets have escaped uninvestigated. This bound is tightened, with increased computation, by including in the bound all targets which will escape uninvestigated even if investigated next. This rule is used in the algorithm presented in Section IV,E.

For the case when the label of the last node in the branch is negative, the lower bound corresponds to the number of targets to escape if the current path is followed and all targets corresponding to nodes at the end of the branch with negative labels are <u>not</u> investigated next. This bound is particularly difficult to compute in that it must take into account all feasible solutions for which the beginning

portion of the permutation corresponds to the current path, excluding those having as their next element an index corresponding to one of the negative labels just described. A method of computing the bound which is not problem oriented is to compute the bound as described in the preceding paragraph associated with the current path as extended by each eligible target and select the minimum of these as a bound on the negatively labeled node. An obvious acceleration is provided by computing these bounds sequentially, keeping track of the minimum bound computed to date, and terminating computation of the bound for a target when it is found to equal the current minimum. This procedure was used in experimental testing.

D. PROBLEMS WHICH CAN BE SOLVED USING THE ALGORITHM

1. Discussion

The algorithm stated in the preceding section is really the skeleton of an algorithm. This basic structure must be augmented with branching rules and methods for computing bounds consistent with the problem formulation. The selection of branching rules is not difficult and can even be done experimentally. However, the determination of usable methods for computing bounds is usually difficult.

The technique is essentially a "one step, look ahead" scheme in which a target is used to extend some branch, and the results of this extension are analyzed and used as a bound on the path represented by the branch. The

look ahead portion, which corresponds to computing the bound, is carried out by computing the time at which investigation of all targets in the newly extended path is complete and determining the number of targets lost as a result of this extension.

The determination of whether an algorithm can be developed for a particular problem formulation depends on whether meaningful bounds can be computed. The efficiency of any such algorithm depends heavily upon the ease with which these bounds are computed.

2. Determination of Suitability of a Formulation for Application of the Algorithm

$I_j(t_i)$, the time required to investigate target j given that investigation of target i is complete at time t_i may be called for many times during the computation of one bound. There may be no known technique for carrying out this computation due either to the complexity or uncertainty of the motion involved - or, on the other hand, it may be the case that the motion renders the functional form trivial. In any case, this is the point where the suitability of the formulation is determined. Even though there may be no standard technique for computing $I_j(t_i)$ which is computationally acceptable, it may be possible to develop new techniques through understanding and exploitation of the particular type of motion involved.

Development of an acceptable functional form for $I_j(t_i)$ is strictly problem oriented and will thus be

illustrated by example. In the first example, the motion described is such that $I_j(t_i)$ takes on a very simple form. In the second, a careful analysis is necessary to determine whether the formulation is suitable, after which special characteristics of the motion involved are noticed and exploited in the development of a surprisingly simple functional form. In both examples it is assumed that an investigation is complete when the position of the target is reached, all targets are available at problem time zero, the investigator changes course instantaneously, and that the investigator's speed is upper bounded by s_1.

a. Same course and speed. Consider the case where all targets have the same course and speed and are proceeding toward the border. Obviously it is best for the investigator to always proceed at maximum speed in this case. Investigation times are thus sequence dependent travel times, independent of start time, illustrating that $I_j(t_i)$ is simply the constant I_{ij} for $i = 1,\ldots,n$, $j = 2,\ldots,n$, which can be precomputed prior to application of the algorithm.

b. Varying courses and speeds. Consider the case where for each i, each coordinate of target i's position is represented parametrically as a function of time. For simplicity, assume the region to be planar and let the position of target i be represented by the couple $(x_i(t),y_i(t))$. Let s_1 be greater than or equal to the maximum speed of all targets.

It is not clear that a functional form for $I_j(t_i)$ can be specified unless an optimal policy for investigator speed can be determined. The following theorem specifies the required policy.

Theorem 4.1: Let the speed of the investigator, $s_1(t)$, be restricted to the interval $(0,s_1)$, i.e. $0 \leq s_1(t) \leq s_1$. Let targets have speeds $s_i(t)$, $i = 2,3,\ldots,n$.

If $s_1 \geq s_i(t)$, $i = 2,3,\ldots,n$, $t \geq 0$, then there exists an optimal solution in which the investigator always uses speed s_1.

Proof: Let π^* be an optimal solution containing n targets in which, on some portion of the path described by π^*, say between targets π_i and π_{i+1}, speed $s_1(t) < s_1$ was used. It will constructively be shown that π^* remains optimal if speed s_1 is used throughout.

Since π^* is optimal, $t_{\pi_j}^* \leq e_{\pi_j}^*$ for all π_j^* in π^*. Note that $t_{\pi_j}^*$ for $j = 1,\ldots,i$ are unaffected by a speed increase between targets π_i and π_{i+1}, but t_{π_j} for $j = i+1,\ldots,n$ may be. Let $\bar{t}_{\pi_j}$ be the completion times after the speed increase for $j = i+1,\ldots,n$. Consider Figure 4.4. Positions A and B represent arbitrary locations of targets π_i and π_{i+1} at time $t_{\pi_i}^*$ and path (B,D) represents the motion of target π_{i+1} during the period $t_{\pi_i}^*$ to $t_{\pi_{i+1}}^*$. (A,D) represents the assumed path followed by the investigator using speed $s_1(t) < s_1$. If speed s_1 were used, some other path such as (A,C) is possible allowing intercept at time

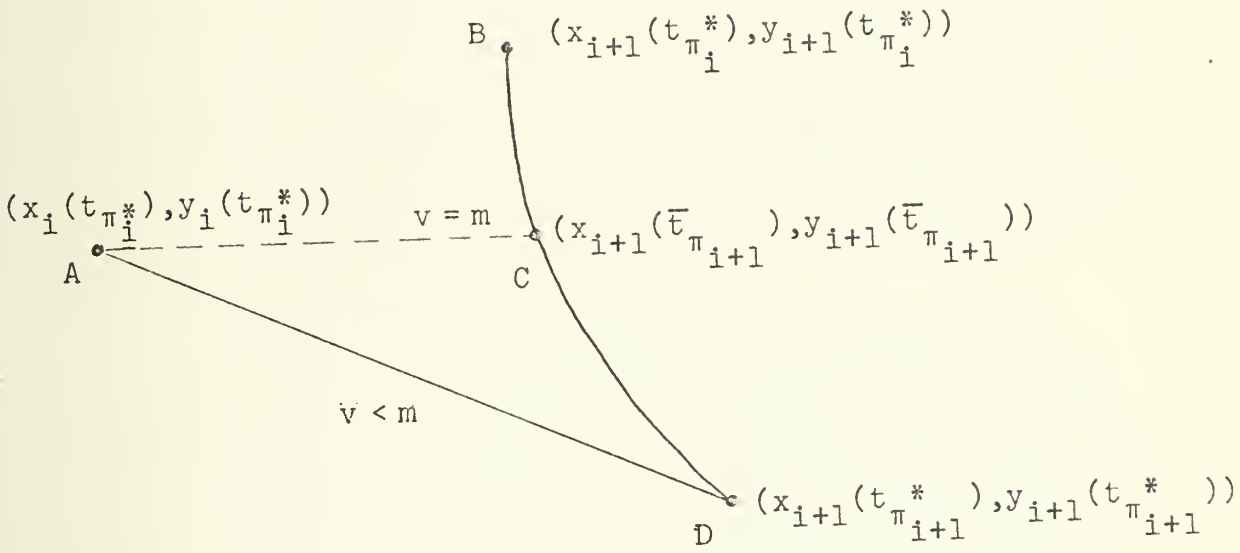


Figure 4.4

$\bar{t}_{\pi_{i+1}} < t_{\pi_{i+1}^*}$. Since s_1 is greater than or equal to the speed of target π_{i+1}, the investigator could proceed along path (C,D), arriving at D at some time t prior to $t_{\pi_{i+1}^*}$, and is free to proceed $t_{\pi_{i+1}^*} - t$ time units earlier from position D to investigate target π_{i+2}. The result is that $\bar{t}_{\pi_j} \leq t_{\pi_j^*}$ for $j = i+1,\ldots,n$ after the speed increase, and since $t_{\pi_j^*} \leq e_{\pi_j}^*$ for $j = i+1,\ldots,n$ π^* remains feasible and thus optimal.

If there were more than one portion of the path on which less than maximum speed were used, the same

reasoning shows that only improvement is the result of increasing speed to s_1 on all such portions.

As a consequence of the theorem, it follows that there exists an optimal solution in which the investigator travels throughout at speed s_1, turning only when an investigation is made. If $s_1 < s_i(t)$ for some i, then this is not necessarily true. In such cases $s_1(t)$, $t \geq 0$, becomes a decision variable, and a solution consists not only of the permutation $\pi=(\pi_1,\ldots,\pi_m)$, but also the speed function $s_1(t)$ for $0 \leq t \leq t_{\pi_m}$.

Under the conditions of the theorem, it is clearly always best for the investigator to proceed from point of departure to point of intercept in a straight line. Hence upon departing the location of target i at time t_i, $(x_i(t_i),y_i(t_i))$, the investigator will proceed away from that point at maximum speed s_1, and thus will be located somewhere on the circumference of a circle centered at $(x_i(t_i),y_i(t_i))$ with radius ts_1 where t is the time elapsed since departure. Thus the investigator's position can be represented as (x,y) where

$$(x_i(t_i)-x)^2 + (y_i(t_i)-y)^2 = (ts_1)^2 \qquad (4\text{-}1)$$

for some $t \geq 0$. The desired time, $I_j(t_i)$, is the minimum time at which the positions of the investigator and target j coincide. Target j's position as a function of elapsed time t is $(x_j(t_i+t),y_j(t_i+t))$. Substitution of these

coordinates into (4-1) in place of (x,y) yields

$$(x_i(t_i)-x_j(t_i+t))^2 + (y_i(t_i)-y_j(t_i+t))^2 = (ts_1)^2. \quad (4-2)$$

Equation (4-2) describes the condition which must be met if the investigator's position and that of target j are to be coincident. The minimum real root of (4-2) is the desired value $I_j(t_i)$.

If s_1 were not greater than the speed of all targets it would be possible for (4-2) to have no real roots. This would indicate that it is not possible for the investigator to carry out the specified investigation.

Problems in which targets have varying courses and speeds have been solved and the solution for one with 40 targets is presented in Section IV,E.

3. Types of Motion Limiting the Applicability of the Algorithm

The suitability of a formulation for application of the algorithm is dependent upon the ability to compute $I_j(t_i)$ efficiently, which in turn depends upon the nature of target and investigator motion. The complexity of functional (4-2) is not apparent since functions $x_i(t)$ and $y_i(t)$ have not been described. The most general motion which can be handled is specified by determining what subset of all functions can be used to represent $x_i(t)$ and $y_i(t)$ and still expect the algorithm to produce an optimal solution in reasonable time. A large subset of functions which

can be considered separately is P, the set of all polynomials. If the positions of targets 2,...,n are expressed as polynomials $p(t) \in P$ of degree two or less, then (4-2) is at most a quartic, for which efficient closed form root extraction techniques exist [Ref. 34]. For those elements of P of degree greater than two, numerical methods could be applied, but the efficiency of these methods would render the algorithm impractical for large problems.

It is interesting to note that if the speed of the investigator is given as a function of time, s(t), then (4-2) becomes

$$(x_i(t_i)-x_j(t_i+t))^2 + (y_i(t_i)-y_j(t_i+t))^2 = (t \int_{t_i}^{t_i+t} s(z)dz)^2. \qquad (4-3)$$

This generalization has application when considering job-shop scheduling problems, in that variations in machine speeds can be handled. For such problems, (4-3) reduces to

$$p_{ij} = t \int_{t_i}^{t_i+t} s(z)dz \qquad (4-4)$$

where p_{ij} is the processing time for job j following job i if the machine is run at normal speed. For example, if machine speed remains normal for some time $\hat{t}$ and then begins to diminish due to normal wear, s(t) would be as shown in Figure 4.5. It may be that the machine is run at normal speed during the day shift, at some lesser speed during the night shift. Such a case is depicted in Figure 4.6.

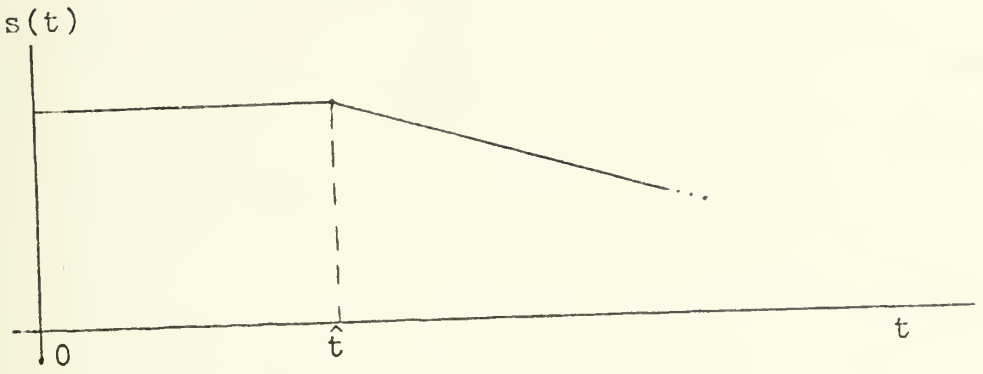


Figure 4.5 Diminishing Machine Speed

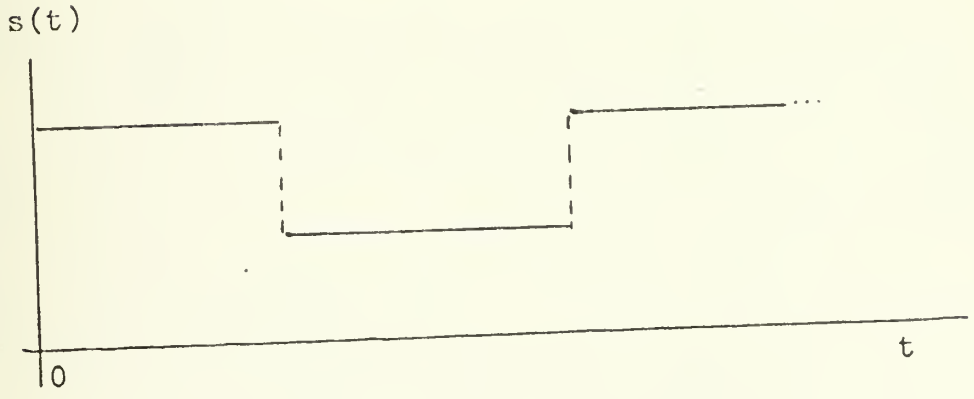


Figure 4.6 Alternating Machine Speed

E. AN ALGORITHM FOR TARGETS WITH DIFFERENT COURSES AND SPEEDS

This section presents an application of Sections IV,C and IV,D to a particular problem.

1. Problem Description

An algorithm has been developed for the problem described in Section II,A and is exhibited in the following section. The motion of target i, i = 2,...,n is represented by a velocity vector $(v_x(i),v_y(i))$. The position of each target at problem time t is $(x_i(t),y_i(t))$ where

$$x_i(t) = x_i^0 + tv_x(i)$$

$$y_i(t) = y_i^0 + tv_y(i)$$

where (x_i^0,y_i^0) is the position of target i at problem time zero. Assuming the investigator to have speed capability greater than or equal to that of any target, the appropriate functional form for $I_j(t_i)$ is given by equation (4-2).

The branch selected for extension is the one with minimum bound. If there is a tie, the branch with a positive label is chosen. If both tied nodes have negative labels, the last node created is used. The rule used for selection of a target to extend the chosen path is the SDP rule discussed in Section IV,B. Bounding procedures are described in Section IV,C.

The border is that portion of the x-axis described by the interval [0,R].

The algorithm for this problem was coded in Fortran IV and run in experimental testing on an IBM 360. The following section contains the flow diagram of the algorithm,

followed by an example problem along with its solution.

A brief summary of computational experience is given.

2. Flow Diagram

The following notation is used:

$(x_i(t), y_i(t))$ = position of target i at time t.

$(v_x(i), v_y(i))$ = velocity vector for target i.

LN = sequence number of last node created in the branching tree.

TN(i) = the label, or target number, associated with node i.

PRED(i) = the node immediately preceding node i in the branching tree.

BB(i) = the bound on node i.

B(i) = BB(i) if node i is currently at the end of a branch in the tree.
M otherwise.

T(i) = the time at which investigation of all targets in the path specified by node i is complete.

INV = the target number specifying the location of the investigator at the current iteration.

MN = the node number of the last positively labeled node in the branch being extended. (Note that TN(MN) = INV.)

BFN = the node at the end of the branch being extended.

BTT = the target selected to extend the current path.

IE(i) = 1 if target i is eligible for use in extending the current path at iteration one.
0 otherwise. (M large.)

E(i) = 1 if target i is eligible for use in extending the current path.

0 otherwise.

TE(i) = E(i) except that TE(i) = 0 for all targets i corresponding to negatively labeled nodes at the end of the branch being extended.

The following is a brief description of the computation carried out in each section of the flow diagram shown on the pages that follow:

a. Ⓐ to Ⓑ. Create node i and set IE(i) = 0 for all targets i whose motion will not take them across the border.

b. Ⓑ to Ⓒ. Compute the initial bound on node i and refine the IE vector.

c. Ⓒ to Ⓓ. Determine MN and the TE vector.

d. Ⓓ to Ⓔ. Select target BTT for use in extending the current path. BTT is the eligible target currently closest to the investigator. Next is the second closest.

e. Ⓔ to Ⓕ. Compute the bound on node labeled BTT.

f. Ⓕ to Ⓖ. Compute bound for NEXT.

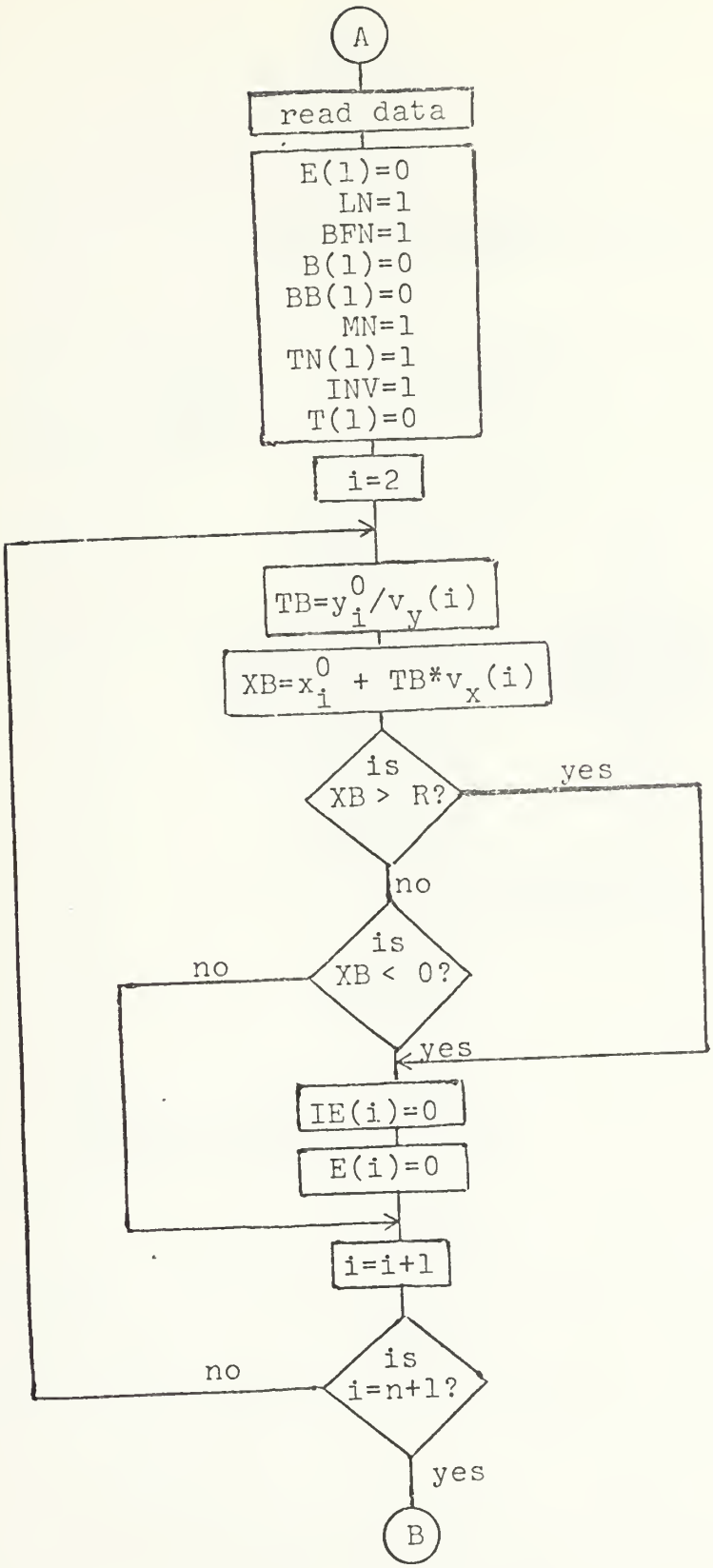
g. Ⓖ to Ⓙ. Sequentially begin to compute bounds for all other eligible targets, keeping track of the minimum which initially is the bound for NEXT, and terminating computation when the bound for a target is found to be equal to the minimum.

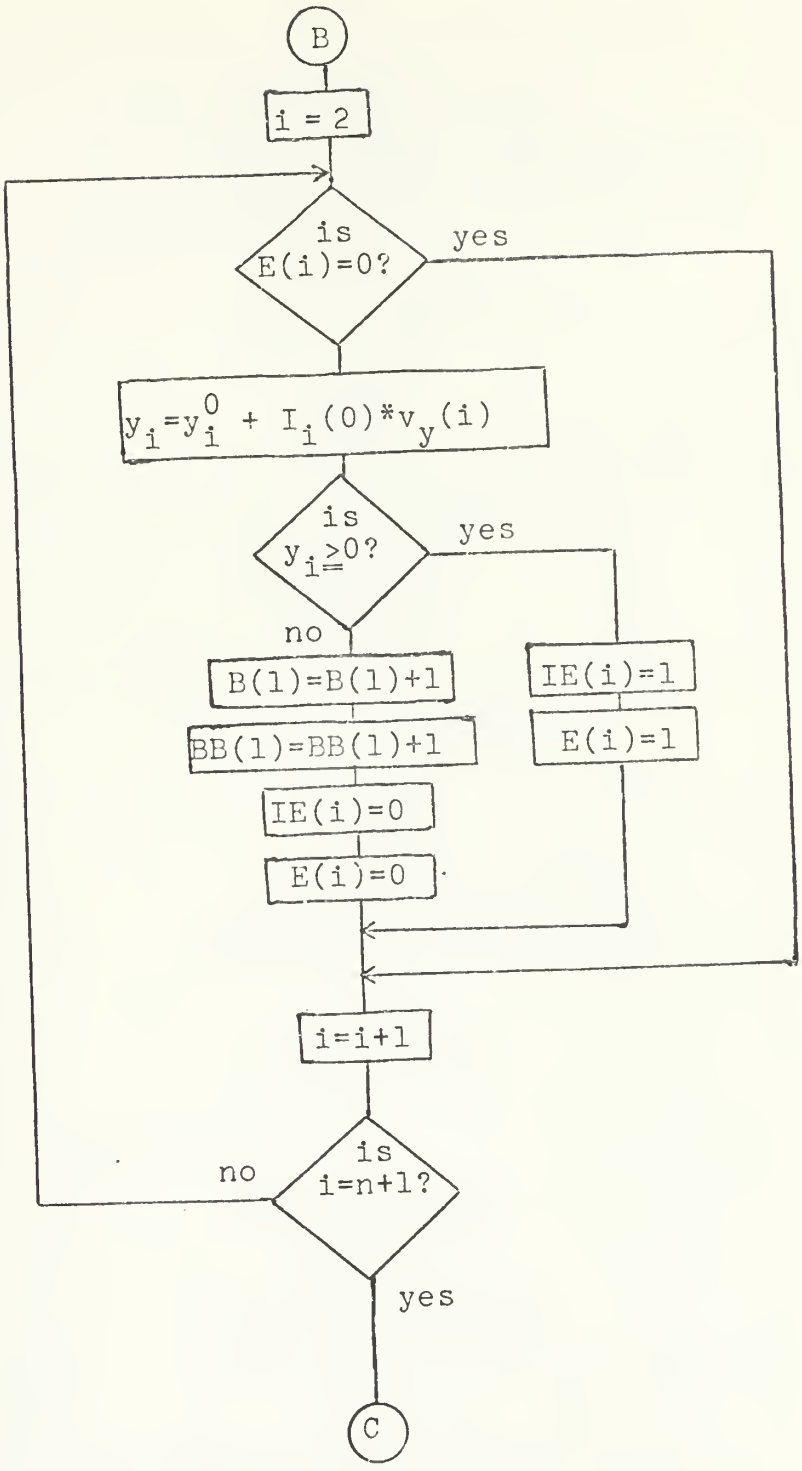
h. Ⓙ to Ⓚ. Create new node labeled -BTT.

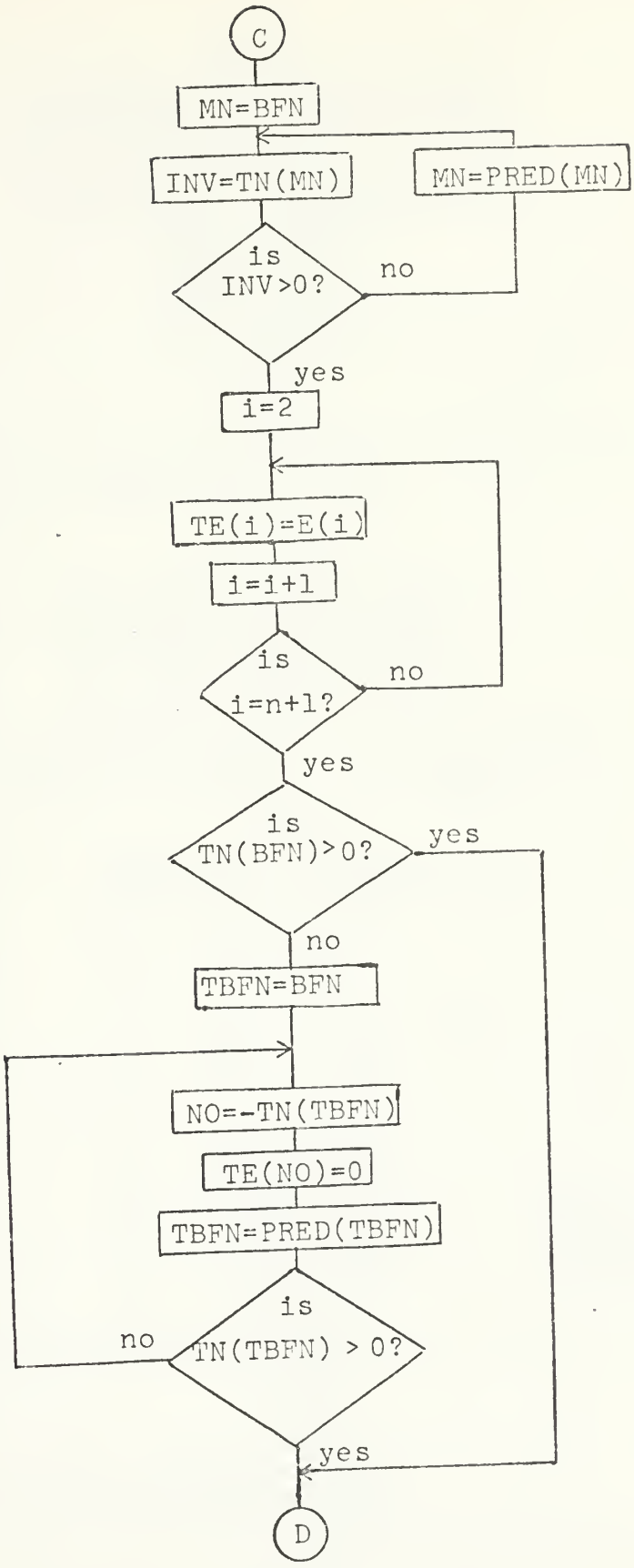
i. Ⓚ to Ⓛ. Select the branch to extend next.

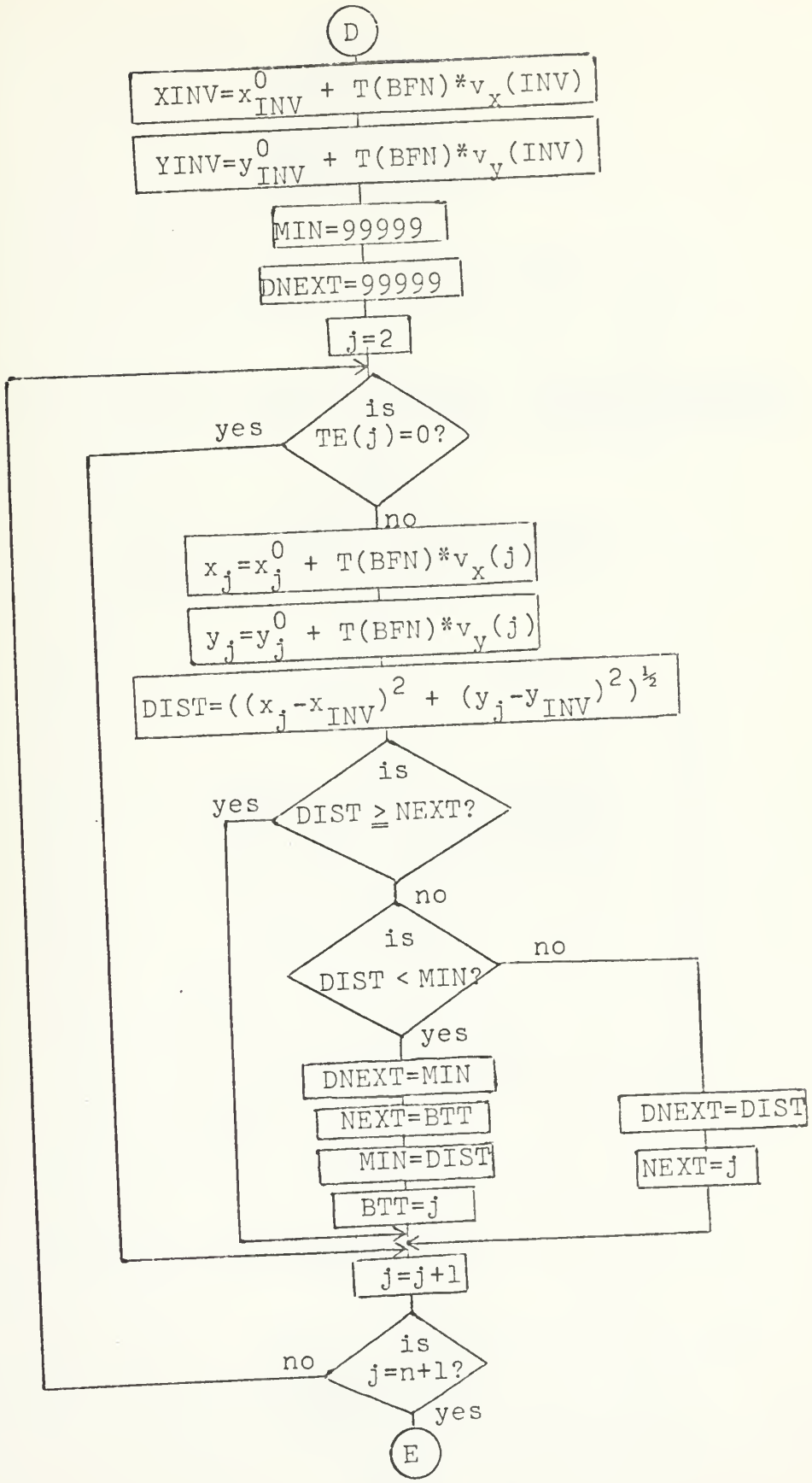
j. Ⓛ to Ⓝ. Determine the set of eligible targets at the current iteration.

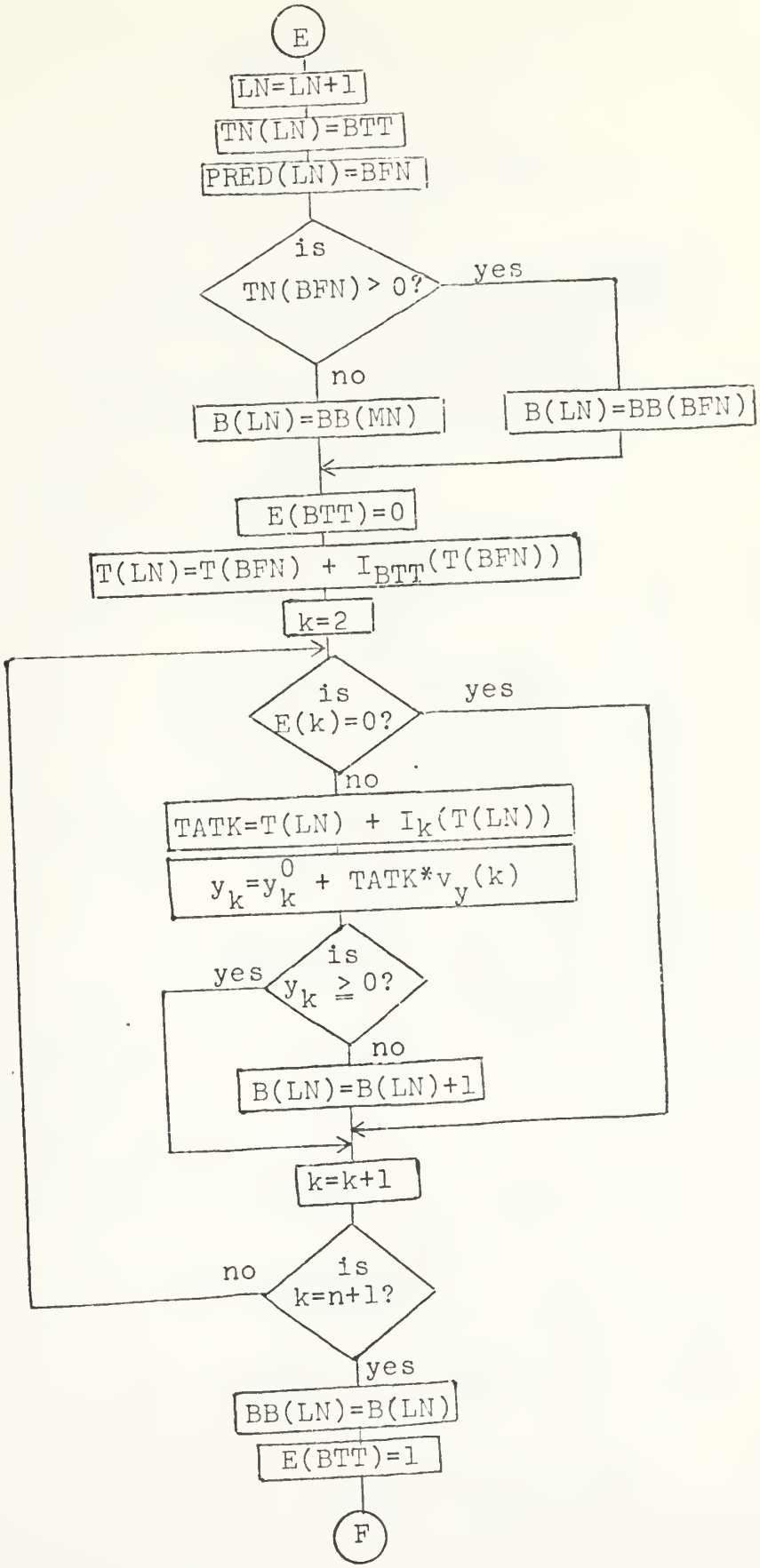
k. Ⓝ to END. Check for termination and prepare to print out solution.

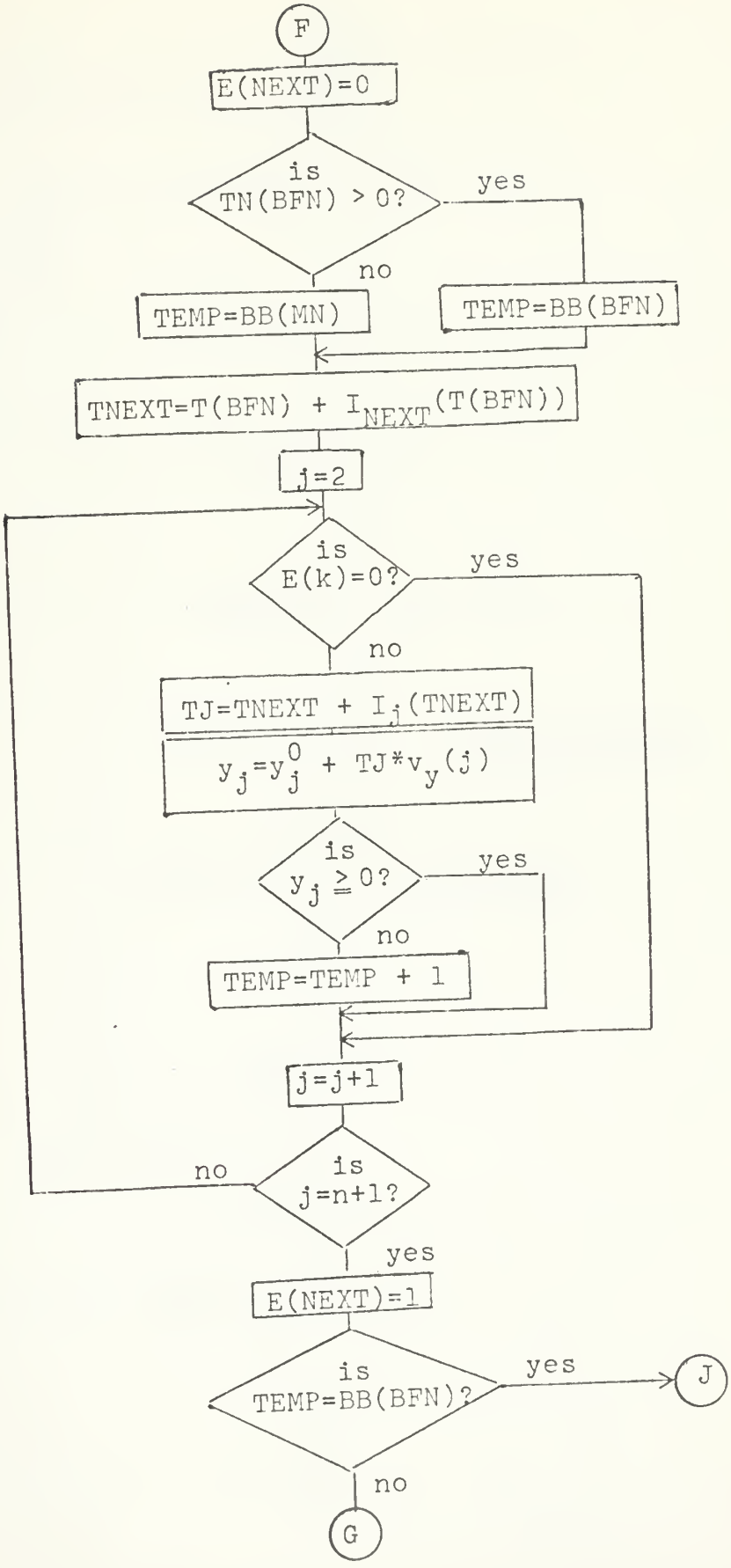

A
read data
E(1)=0
LN=1
BFN=1
B(1)=0
BB(1)=0
MN=1
TN(1)=1
INV=1
T(1)=0
i=2
TB=y_i^0/v_y(i)
XB=x_i^0 + TB*v_x(i)
is XB > R?
yes
no
is XB < 0?
no
yes
IE(i)=0
E(i)=0
i=i+1
is i=n+1?
no
yes
B

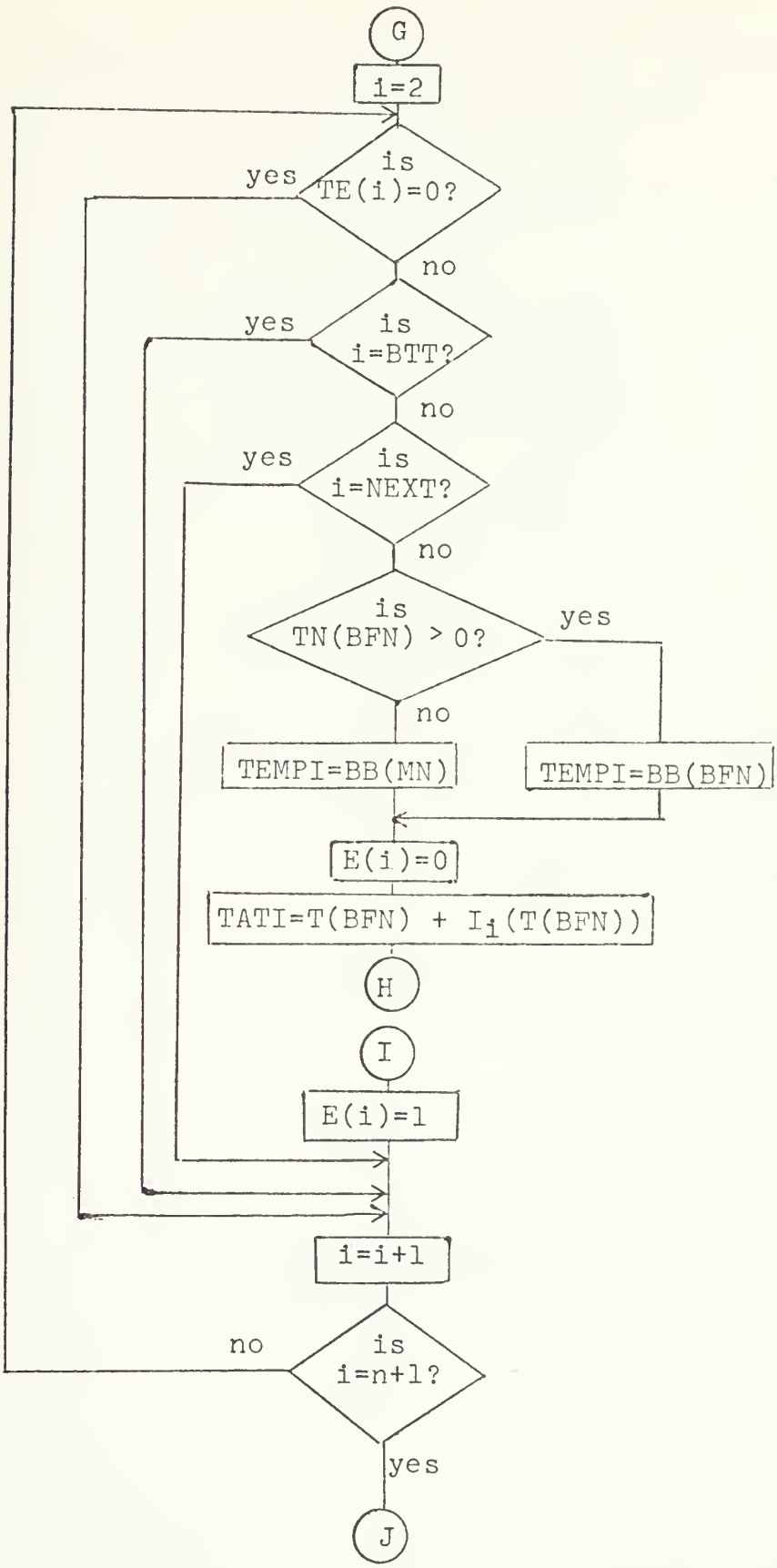

B
i = 2
is E(i)=0?
yes
$y_i=y_i^0 + I_i(0)*v_y(i)$
is $y_i \geq 0$?
yes
no
B(1)=B(1)+1
BB(1)=BB(1)+1
IE(i)=0
E(i)=0
IE(i)=1
E(i)=1
i=i+1
is i=n+1?
no
yes
C

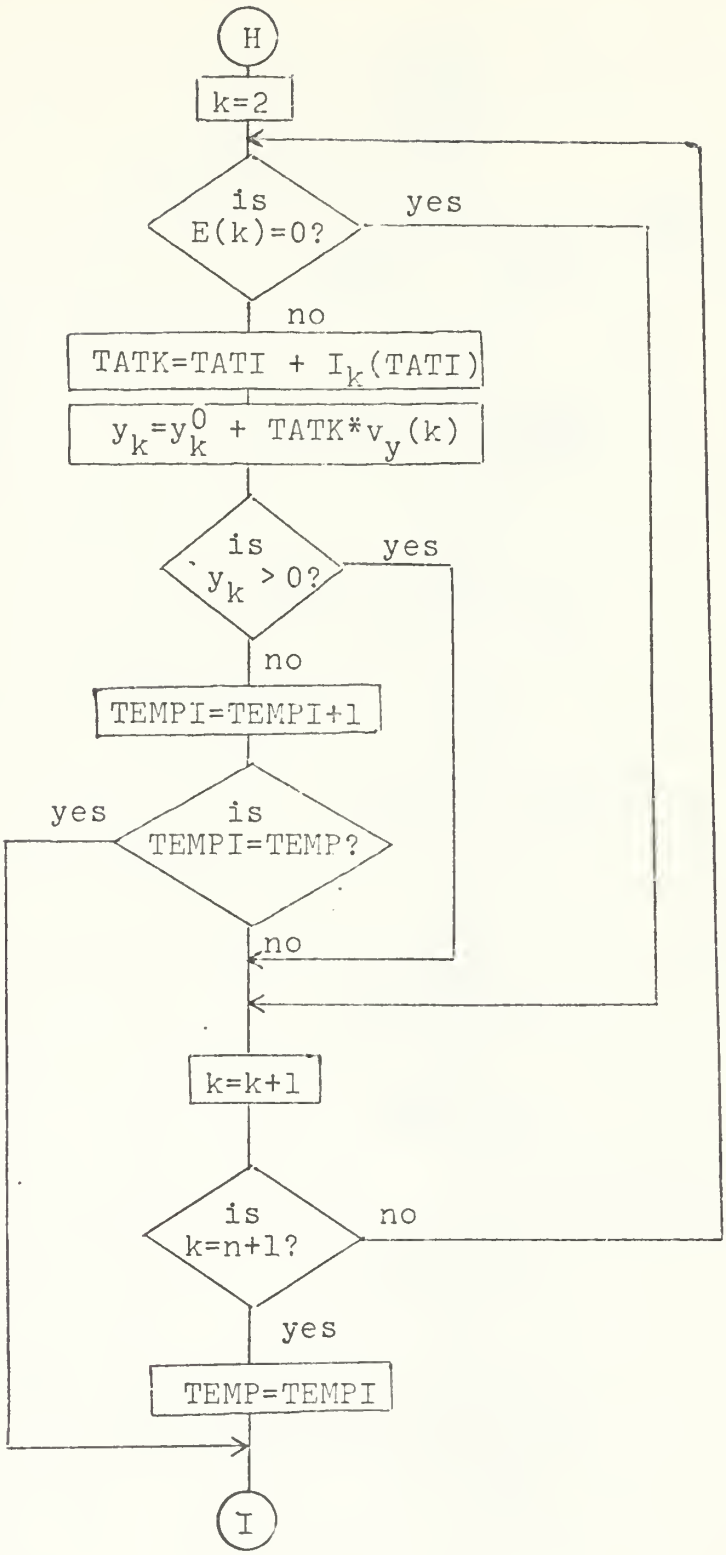

C
MN=BFN
INV=TN(MN)
MN=PRED(MN)
is INV>0?
no
yes
i=2
TE(i)=E(i)
i=i+1
is i=n+1?
no
yes
is TN(BFN)>0?
yes
no
TBFN=BFN
NO=-TN(TBFN)
TE(NO)=0
TBFN=PRED(TBFN)
is TN(TBFN) > 0?
no
yes
D

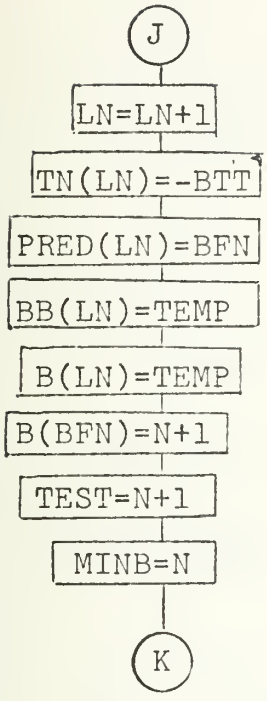

D
XINV=x^0_{INV} + T(BFN)*v_x(INV)
YINV=y^0_{INV} + T(BFN)*v_y(INV)
MIN=99999
DNEXT=99999
j=2
is TE(j)=0?
yes
no
$x_j=x^0_j$ + T(BFN)*v_x(j)
$y_j=y^0_j$ + T(BFN)*v_y(j)
DIST=$((x_j-x_{INV})^2 + (y_j-y_{INV})^2)^{1/2}$
is DIST ≥ NEXT?
yes
no
is DIST < MIN?
no
yes
DNEXT=MIN
NEXT=BTT
MIN=DIST
BTT=j
DNEXT=DIST
NEXT=j
j=j+1
is j=n+1?
no
yes
E

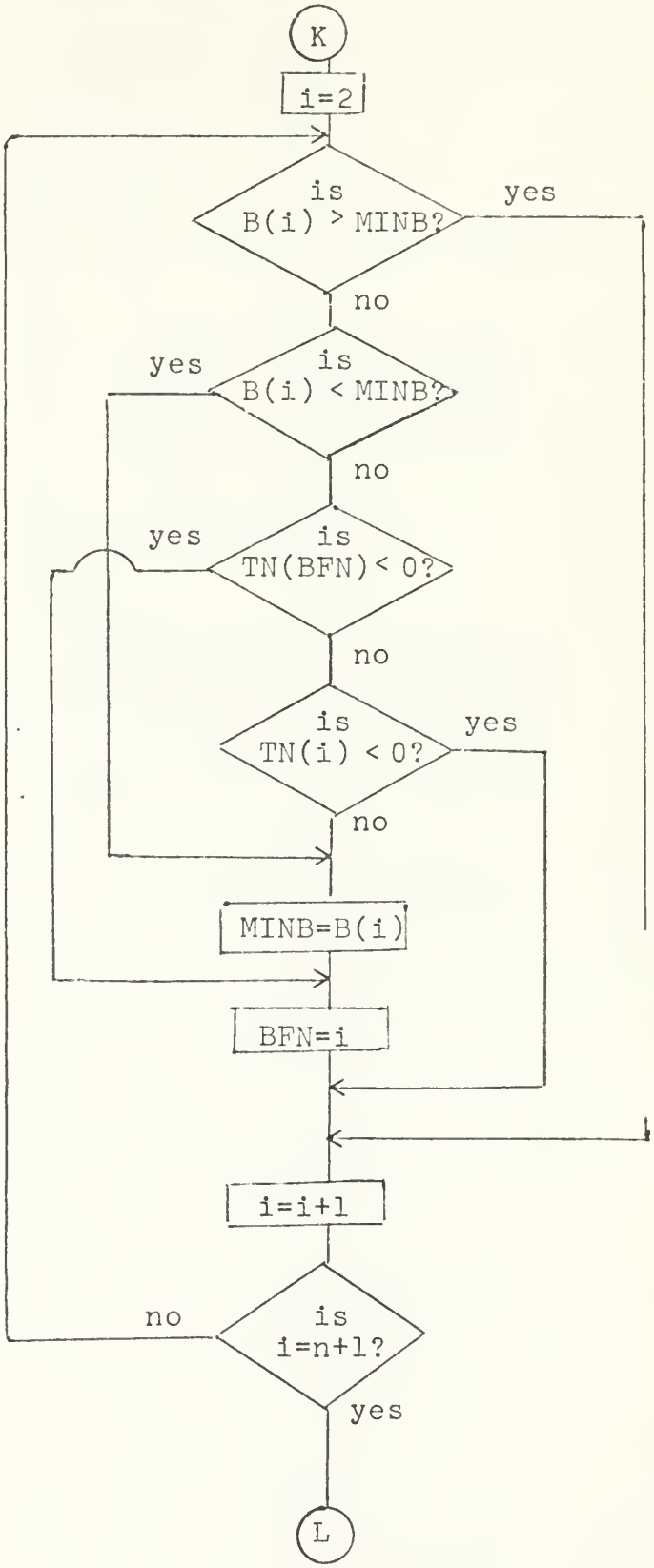

E
LN=LN+1
TN(LN)=BTT
PRED(LN)=BFN
is TN(BFN) > 0?
yes
no
B(LN)=BB(MN)
B(LN)=BB(BFN)
E(BTT)=0
T(LN)=T(BFN) + I_BTT(T(BFN))
k=2
is E(k)=0?
yes
no
TATK=T(LN) + I_k(T(LN))
y_k=y_k^0 + TATK*v_y(k)
is y_k ≥ 0?
yes
no
B(LN)=B(LN)+1
k=k+1
is k=n+1?
no
yes
BB(LN)=B(LN)
E(BTT)=1
F

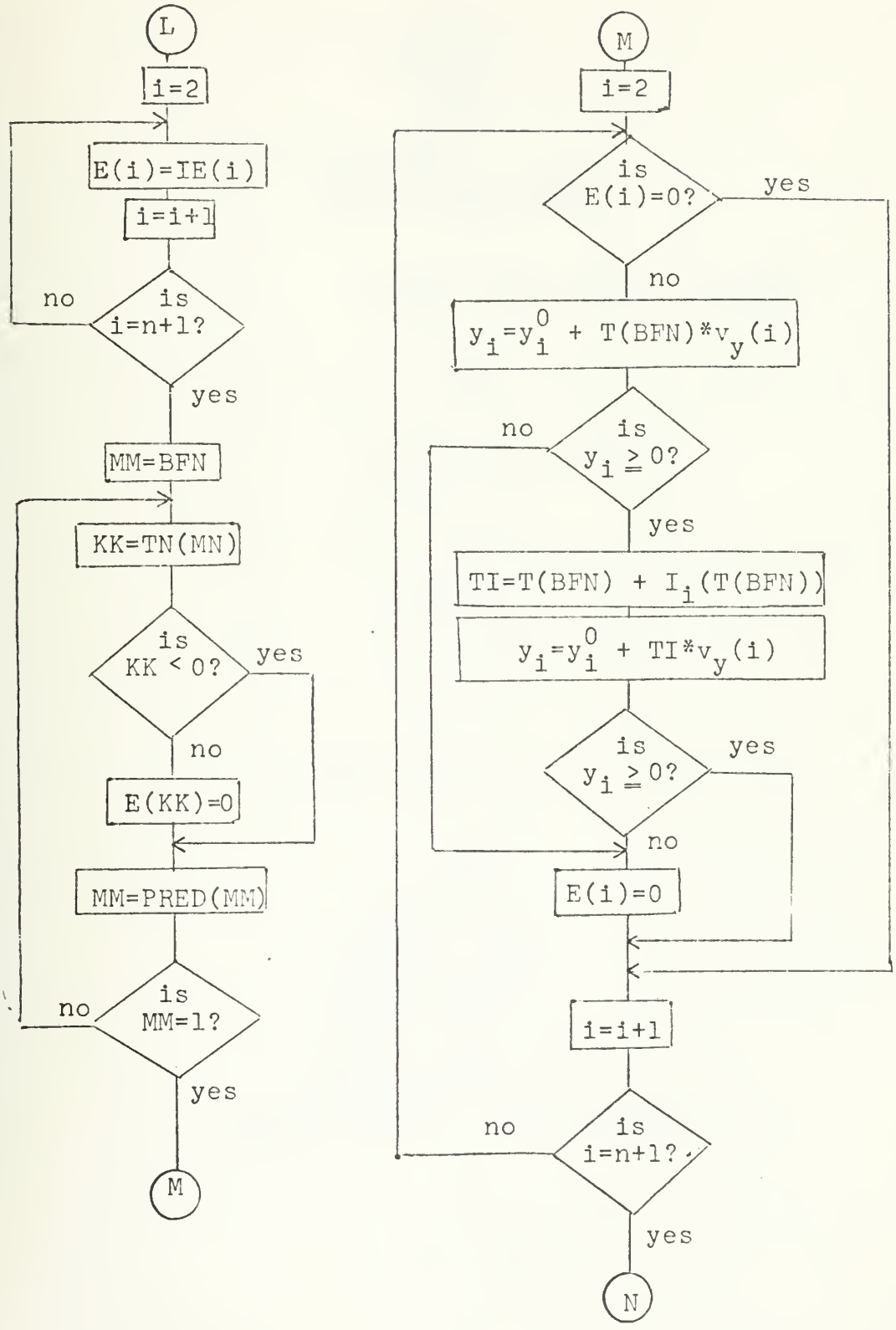

F
E(NEXT)=0
is TN(BFN) > 0?
yes
no
TEMP=BB(MN)
TEMP=BB(BFN)
TNEXT=T(BFN) + I_NEXT(T(BFN))
j=2
is E(k)=0?
yes
no
TJ=TNEXT + I_j(TNEXT)
y_j=y_j^0 + TJ*v_y(j)
is y_j ≥ 0?
yes
no
TEMP=TEMP + 1
j=j+1
is j=n+1?
no
yes
E(NEXT)=1
is TEMP=BB(BFN)?
yes
J
no
G

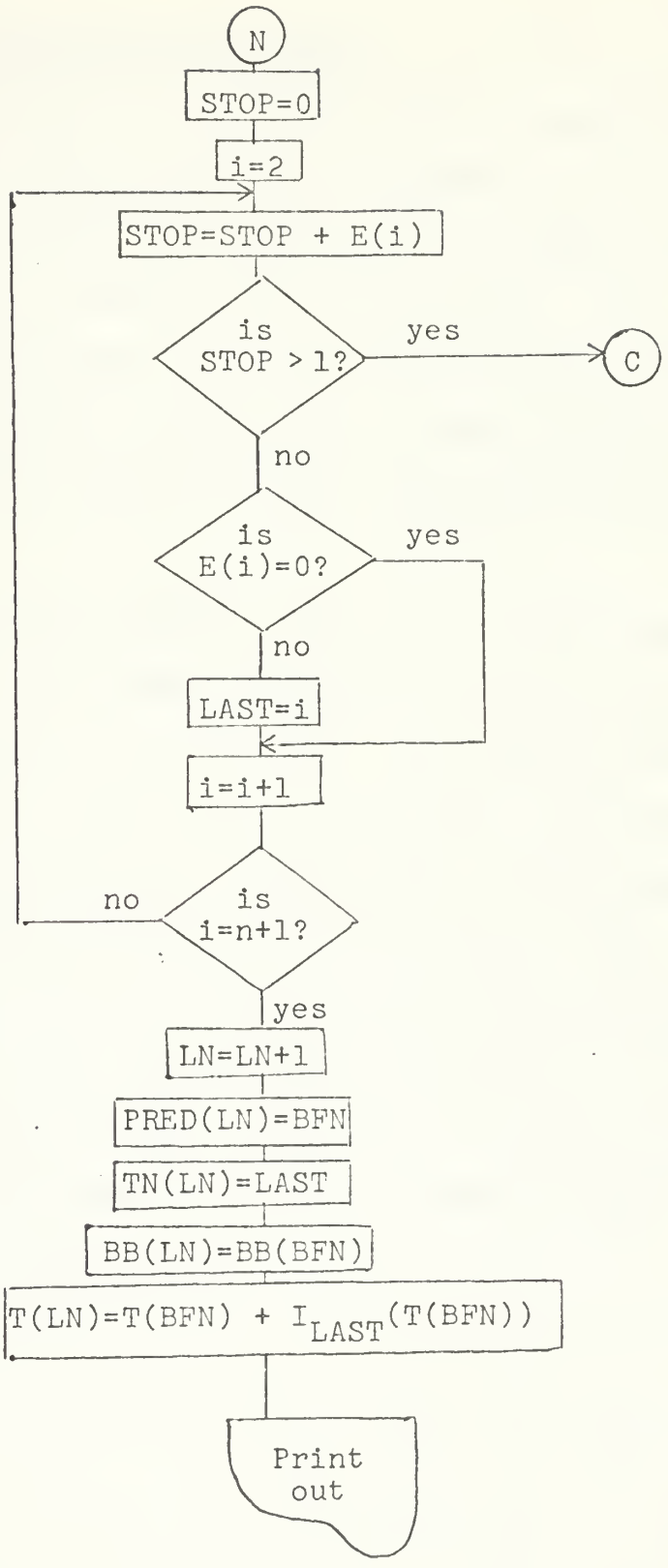

G
i=2
is TE(i)=0?
yes
no
is i=BTT?
yes
no
is i=NEXT?
yes
no
is TN(BFN) > 0?
yes
no
TEMPI=BB(MN)
TEMPI=BB(BFN)
E(i)=0
TATI=T(BFN) + I_i(T(BFN))
H
I
E(i)=1
i=i+1
is i=n+1?
no
yes
J


H
k=2
is E(k)=0?
yes
no
TATK=TATI + I_k(TATI)
$y_k=y_k^0$ + TATK*v_y(k)
is y_k > 0?
yes
no
TEMPI=TEMPI+1
yes
is TEMPI=TEMP?
no
k=k+1
is k=n+1?
no
yes
TEMP=TEMPI
I


J
LN=LN+1
TN(LN)=-BTT
PRED(LN)=BFN
BB(LN)=TEMP
B(LN)=TEMP
B(BFN)=N+1
TEST=N+1
MINB=N
K


K
i=2
is B(i) > MINB?
yes
no
is B(i) < MINB?
yes
no
is TN(BFN) < 0?
yes
no
is TN(i) < 0?
yes
no
MINB=B(i)
BFN=i
i=i+1
is i=n+1?
no
yes
L


L
i=2
E(i)=IE(i)
i=i+1
is i=n+1?
no
yes
MM=BFN
KK=TN(MN)
is KK < 0?
yes
no
E(KK)=0
MM=PRED(MM)
is MM=1?
no
yes
M
M
i=2
is E(i)=0?
yes
no
$y_i=y_i^0 + T(BFN)*v_y(i)$
is $y_i \geq 0$?
no
yes
$TI=T(BFN) + I_i(T(BFN))$
$y_i=y_i^0 + TI*v_y(i)$
is $y_i \geq 0$?
yes
no
E(i)=0
i=i+1
is i=n+1?
no
yes
N


N
STOP=0
i=2
STOP=STOP + E(i)
is STOP > 1?
yes
C
no
is E(i)=0?
yes
no
LAST=i
i=i+1
no
is i=n+1?
yes
LN=LN+1
PRED(LN)=BFN
TN(LN)=LAST
BB(LN)=BB(BFN)
T(LN)=T(BFN) + I_LAST(T(BFN))
Print out

3. Sample Problem and Solution

This algorithm was used to develop optimal solutions for problem sets with as many as 50 targets initially in the region. The solution obtained for one sample problem with 40 targets is presented for illustration. The initial positions and velocities for the 40 targets were selected randomly from uniform distributions, target speeds ranging from 10 to 20 units per unit time. The data is contained in Table 4.3.

Only those 25 targets whose motion would take them across the border are shown in Figure 4.7, their initial locations being denoted by circles. The optimal path for the investigator is shown with a solid line. The motion of all targets is shown with dashed lines. Heavy dots show the locations where investigations take place. The optimal solution is (1,2,6,4,8,21,13,27,16,23,25,34,20,15,35,28,37).

A set of seven such problems was solved on an IBM 360. CPU time ranged from 2.46 to 78.27 seconds, averaging 34.91 seconds. The number of bounds computed ranged from 102 to 2190 and averaged 1251.

TARGET	X^o	Y^o	v_x	v_y	TARGET	X^o	Y^o	v_x	v_y
1	6.51	.49	Max Speed = 25		21	6.56	4.97	.6	-18.7
2	6.12	.65	12.5	- 7.95	22	9.60	5.00	- 9.6	- 2.7
3	.41	.84	10.9	- 7.2	23	9.23	5.05	- .9	-11.7
4	6.25	1.84	8.0	- 8.3	24	.39	5.46	-10.2	- 2.1
5	8.16	1.89	3.1	-16.8	25	5.51	5.52	5.78	- 8.8
6	6.47	1.92	15.0	-13.0	26	7.29	5.55	-12.4	- 5.2
7	.64	1.94	14.2	- 8.5	27	5.48	5.75	.9	-19.1
8	9.07	2.68	-18.7	- 7.0	28	.55	6.62	8.6	- 8.2
9	1.55	2.96	- 2.2	-13.9	29	2.90	6.63	-13.9	- 2.8
10	9.73	3.04	11.7	- 6.8	30	.47	6.70	11.1	- .7
11	4.00	3.74	2.3	-16.0	31	6.53	6.74	18.6	- .1
12	5.20	3.79	-15.3	- 4.9	32	4.37	7.09	- 6.2	-13.7
13	3.19	3.85	15.1	-11.8	33	4.39	7.28	14.6	- 7.5
14	1.50	4.03	- 4.5	-12.6	34	3.10	7.32	7.3	-12.8
15	9.62	4.36	-10.9	- 6.2	35	.37	8.39	5.4	-13.0
16	7.11	4.36	- 0.4	-10.3	36	2.67	9.08	-13.0	- 8.2
17	4.13	4.43	11.2	- 3.3	37	9.63	9.08	- 6.7	-10.5
18	.80	4.54	3.6	-11.9	38	1.64	9.15	- 9.2	- 5.0
19	8.56	4.73	.8	-17.1	39	.98	9.31	14.7	- 4.4
20	.88	4.74	9.8	- 6.4	40	9.32	9.40	-12.7	-11.1

Table 4.3. Example Problem Data

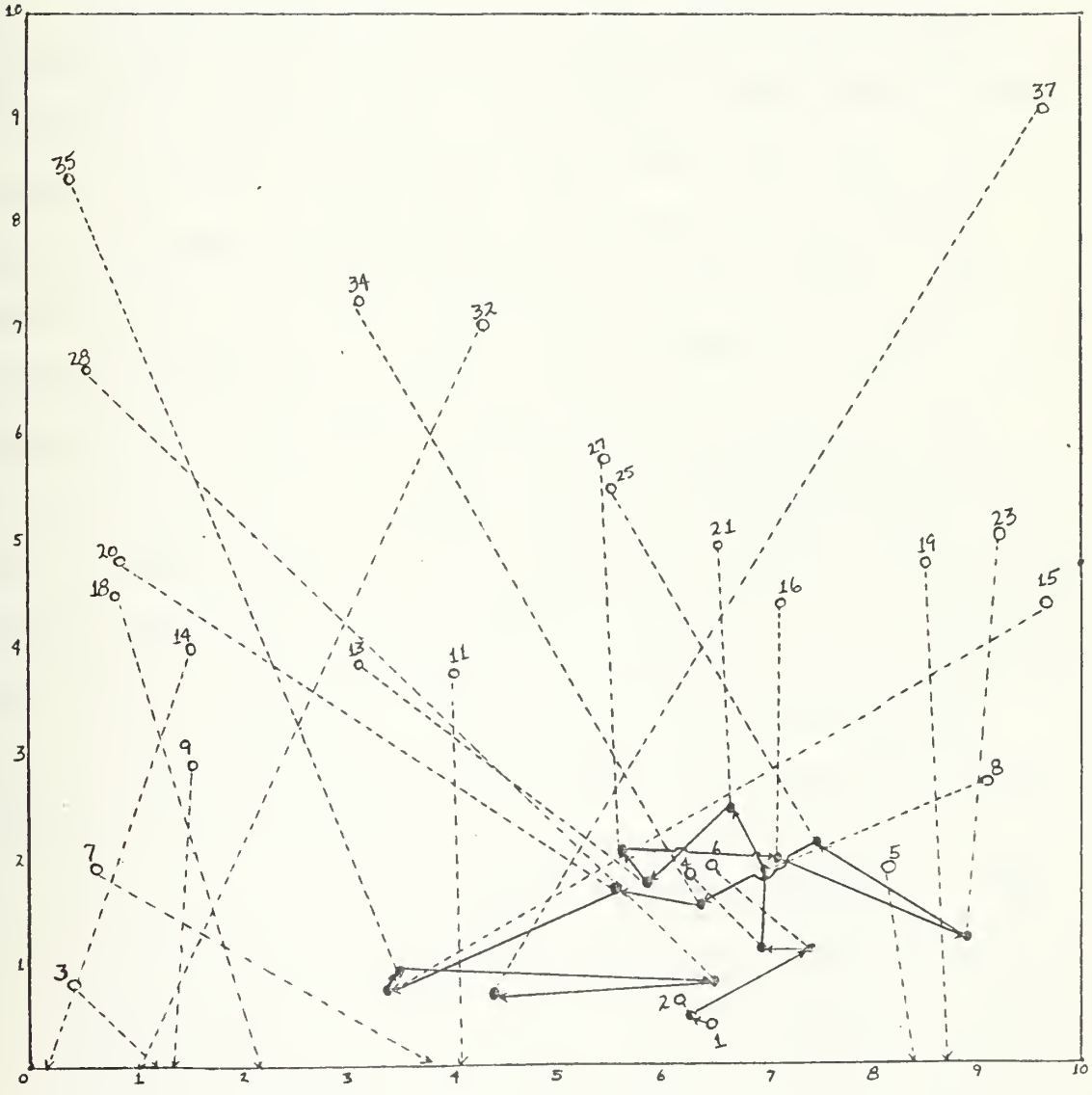


Figure 4.7. Sample Problem and Solution

4. Remarks

Earlier it was noted that the efficiency of an algorithm is directly related to the efficiency with which the value $I_j(t_i)$ could be computed. This can now be illustrated.

The algorithm presented in this section was used to obtain the optimal solutions used in the evaluation of heuristics in Section IV,B. The motion in the first twenty problems considered there specifies that all targets move toward the border at the same speed, and in the second, targets move toward the border at different speeds. For the first problem set, $I_j(t_i)$ is independent of time of initiation of investigation and thus it reduces to I_{ij}, $i = 1,\ldots,n$, $j = 2,\ldots,n$, which were computed and stored prior to application of the algorithm. In the second problem set $I_j(t_i)$ is time dependent and had to be computed each time it was required by the algorithm using a modified form of equation (4-2). A comparison of computational results is shown in Table 4.4. Times are given in seconds.

Motion Type	AVG CPU Time	STD. DEV.	AVG # Bounds	STD DEV.
Same Speed	1.85	1.14	331.9	225.1
Different Speed	13.54	21.9	540.2	709.4

Table 4.4. Comparison of Computational Results

Note that even though the average number of bounds increased only by a factor of 1.6, the average CPU time increased by a factor of 7.3. It is thus clear that the percentage of time spent in carrying out the $I_j(t_i)$ computation is significant. For the second problem set this percentage is conservatively estimated using the data from Table 4.4 to be 60%.

V. UNCERTAIN INVESTIGATION TIMES

A. INTRODUCTORY REMARKS

In all problems considered thus far, investigation times $I_j(t_i)$ were required to be known with certainty. The actual random variation in these times is ignored for the obvious reason of simplification. However, even in the simplest formulations, these variations may be significant enough to warrant the added computational burden involved in accounting for them.

The nature of the variation is usually such that it can be approximated in distribution. It is a common practice in jobshop scheduling to approximate set-up and/or processing times using a normal or related distribution. Similar approaches appear appropriate for investigation times. For example, if targets are ships, travel times between known points will actually vary due to changing wind and sea conditions. It is felt that these times could be closely approximated using a distribution in the normal family, parameters for which could be developed by fitting with an experimentally developed sample distribution.

In order to determine the feasibiliy of a solution to an investigation problem, it is necessary to be able to compute or estimate the time investigation is complete for each target included. Hence, for solution π, it is necessary to compute $t_{[j]}$ for $j=1,\ldots,m$ where

$$t_{[j]} = \sum_{i=0}^{j-1} I_{[i+1]}(t_{[i]}).$$

Note that assuming investigation times to be known only in distribution makes $I_j(t_i)$ a random function for i=1,...,n, j=2,...,n. If the distribution of random variable $I_j(t_i)$ is known and happens to have the reproductive property [Ref. 25] then the distribution of $t_{[j]}$ is also known. The normal random variables may thus be computationally advantageous in that $t_{[j]}$ is again a normal random variable with known distribution.

Under this assumption, it is possible to efficiently solve certain types of investigation problems using the techniques of probabilistic programming. The class of problems which may lend themselves to these techniques includes problems having an integer linear programming formulation in which the number of constraints is not prohibitive. For example, if it is possible to reduce the solution set to subpermutations of a single permutation, as in Section III, then solution techniques for at least moderate size problems may be possible.

There is one group of investigation problems which is known to be solvable using these techniques and an efficient algorithm for obtaining optimal solutions has been developed. The problem is described and solved in the sections which follow. The problem is best understood using the notation and terminology of jobshop scheduling, in that a special case of the problem is well known [Ref. 32].

B. SCHEDULING WITH DUE-DATES AND UNCERTAIN SET-UP AND PROCESSING TIMES

The situation treated is one in which a single machine, (investigator), is used to process a presently available set of jobs, (targets), with known due-dates (escape times). Each job consists of a single operation but set-up and processing times (investigation times) are random with known distributions. Once processing of a job has been initiated, it is processed to completion. The objective is to specify, prior to commencement of any job, a schedule which will minimize the number of late jobs.

If set-up and processing times are known with certainty, Moore's algorithm can be applied to produce the optimal schedule as follows. All n jobs are ordered according to non-decreasing due-dates and partial schedules up to job i in the sequence are considered for i = 1,2,...,n. If job i will be late if this ordering is followed, then from among jobs one through i, that job with the largest processing time is excluded from the schedule. Repeated application of this exclusion rule each time a partial schedule shows that a job must be excluded will produce a schedule in which the number of late jobs is minimized.

In the problem being considered set-up and processing times are assumed to be random. The problem is cast into the form of an integer program and the constraints transformed into chance constraints. A certainty equivalent for Moore's job with the longest processing time is achieved and

a generalization of Moore's Algorithm is presented. The algorithm is then proved to produce an optimal schedule for the deterministic equivalent problem.

1. Integer Programming Formulation

It is assumed throughout that processing times and set-up times are independent normally distributed random variables. The set-up and processing times for each job are summed to form a new random variable p, called simply processing time.

The problem will initially be formulated as an integer program. Let p_i be the processing time for job i, $i = 1,\ldots,n$ and d_i the due date. Let $x_i = 1$ if job i is included in the schedule and 0 otherwise. Jobs are indexed such that $d_1 \leq d_2 \leq \ldots \leq d_n$. Associated with each job is a constraint of the following form which requires, if the job is processed, that it be completed prior to its due date:

$$\sum_{j=1}^{j=i} p_j x_j \leq d_i \qquad i = 1,2,\ldots,n \qquad (5\text{-}1)$$

The objective is to maximize $\sum_{j=1}^{j=n} x_j$, the number of early jobs.

2. Chance Constrained Formulation

As the coefficients in (5-1) are random, these constraints cannot be met with certainty. Application of the technique of Ref. 9 transforms inequalities (5-1) into risk constraints where the degree of risk is determined by

the decision maker. The problem then becomes

$$\text{maximize} \quad \sum_{j=1}^{j=n} x_j$$

$$\text{subject to} \quad P\left[\sum_{j=1}^{j=i} P_j x_j \leq d_i\right] \geq a \ , \quad i = 1,\ldots,n \tag{5-2}$$

Constraints (5-2) now require that whatever schedule be selected, it must satisfy (5-1) with probability at least equal to a.

Note that the left hand side of each constraint in (5-1) is a sum of independent normal random variables. Rewriting a constraint in (5-2) as

$$P\left[d_i - \sum_{j=1}^{j=i} p_j x_j \geq 0\right] \geq a \tag{5-3}$$

and normalizing, the equivalent form

$$\Phi\left((d_i - \sum_{j=1}^{j=i} m_j x_j)/\left(\sum_{j=1}^{j=i} v_j^2 x_j^2\right)^{\frac{1}{2}}\right) \geq a \tag{5-4}$$

is obtained where Φ is the cumulative distribution function for the standard normal random variable and p_j has mean m_j and variance v_j^2. Letting K be the inverse of Φ, constraint (5-4) becomes

$$d_i - \sum_{j=1}^{j=i} m_j x_j \geq K(a)\left(\sum_{j=1}^{j=i} v_j^2 x_j^2\right)^{\frac{1}{2}} . \tag{5-5}$$

Auxiliary spacer variables [Ref. 10] are introduced to obtain two constraints from (5-5) as follows:

$$\sum_{j=1}^{j=i} m_j x_j + y_i \leq d_i \qquad (5\text{-}6)$$

$$-\sum_{j=1}^{j=i} v_j^2 x_j + y_i^2/h \geq 0$$

where $h = (K(a))^2$, x_j^2 is replaced by x_j since being either 0 or 1 it has the same effect, and y_i is the spacer variable.

The final problem is

$$\text{maximize} \quad \sum_{j=1}^{j=n} x_j$$

$$\text{subject to} \quad \sum_{j=1}^{j=i} m_j x_j + y_i \leq d_i \quad i = 1,\ldots,n \qquad (5\text{-}7)$$

$$-\sum_{j=1}^{j=i} v_j^2 x_j + y_i^2/h \geq 0 \quad i = 1,\ldots,n \qquad (5\text{-}8)$$

$$x_j = 0,1 \qquad y_i \geq 0$$

The resulting problem is a nonlinear integer programming problem with n quadratic constraints. Due to the integer nature of the x_j the problem remains intractable, even in this simplified form. It is possible, however, to deduce a great deal about the nature of the solution through inspection of constraints (5-7) and (5-8).

Note that in (5-7) instead of having a job's completion time constrained to be less than its due-date, as it is

in (5-1), the expected completion time plus a risk aversion factor, y_i, is required to be less than the corresponding due date. Note also that (5-8) requires that the risk aversion factor be such that

$$y_i \geq (h \sum_{j=1}^{j=i} v_j^2 x_j)^{\frac{1}{2}} \quad . \tag{5-9}$$

As can be seen by consideration of (5-9), larger variances of jobs in the schedule up to and including job i produce a greater risk aversion factor y_i, and therefore more slack is required in constraint (5-7) over and above the expected processing time.

3. Development of the Deterministic Equivalent

The above observations are exploited and extended to produce a method of arriving at an optimal solution to the transformed problem.

The problem development of sections V,B,1 and V,B.2 effectively states that in order to arrive at an optimal schedule, one should order the jobs according to due-dates, compute the adjusted expected completion times of jobs in ascending order of job index number, and when the first job in the sequence is encountered for which constraint (5-7) is violated, select from among the jobs included in the sequence up to that point one for exclusion from the sequence.

Letting I_i represent the index set for jobs included in the schedule up to job i, when applying Moore's algorithm, a partial schedule is developed until job i is encountered

such that

$$\sum_{j \in I_i} P_j > d_i \quad . \tag{5-10}$$

Moore then selects for exclusion that job whose exclusion minimizes the left side of (5-10), i.e. job k where

$$P_k = \max p_j \ , \quad j \text{ in } I_i \ . \tag{5-11}$$

It is this minimizing characteristic upon which Moore's proof of optimality is based.

The certainty equivalent for Moore's job with the longest processing time is job k identified in the following exclusion rule.

EXCLUSION RULE: In the deterministic equivalent problem, when a partial schedule I_i is developed for which constraint (5-7) is violated, select for exclusion from I_i index k where

$$m_k + h^{\frac{1}{2}} \left[\left(\sum_{j \in I_i} v_j^2 \right)^{\frac{1}{2}} - \left(\sum_{\substack{j \in I_i \\ j \neq k^i}} v_j^2 \right)^{\frac{1}{2}} \right] = \max_{r \in I_i} \left\{ m_r + h^{\frac{1}{2}} \left[\left(\sum_{j \in I_i} v_j^2 \right)^{\frac{1}{2}} - \left(\sum_{\substack{j \in I \\ j \neq r^i}} v_j^2 \right)^{\frac{1}{2}} \right] \right\} \tag{5-12}$$

This is now shown to be true. It is useful to rewrite (5-7) and (5-8) as a single constraint by substituting the right hand side of (5-9) into (5-7) yielding

$$\sum_{j=1}^{j=i} m_j x_j + \left(h \sum_{j=1}^{j=i} v_j^2 x_j\right)^{\frac{1}{2}} \leq d_i \qquad i = 1,\ldots,n \qquad (5\text{-}13)$$

Suppose (5-13) is satisfied for job i-1, but not for job i. Then the following condition exists:

$$\sum_{j=1}^{j=i-1} m_j x_j + \left(h \sum_{j=1}^{j=i-1} v_j^2 x_j\right)^{\frac{1}{2}} \leq d_{i-1} \qquad (5\text{-}14)$$

and

$$\sum_{j=1}^{j=i} m_j x_j + \left(h \sum_{j=1}^{j=i} v_j^2 x_j\right)^{\frac{1}{2}} > d_i \qquad (5\text{-}15)$$

where $x_j=1$ for j in I_i and $x_j=0$ otherwise. If r is in I_i and x_r is set to zero, the left side of (5-15) is reduced by

$$m_r + h^{\frac{1}{2}} \left[\left(\sum_{j \in I_i} v_j^2 \right)^{\frac{1}{2}} - \left(\sum_{\substack{j \in I_i \\ j \neq r^i}} v_j^2 \right)^{\frac{1}{2}} \right] .$$

Setting $x_i=0$ produces a left side in (5-15) less than or equal to d_{i-1} due to (5-14) and $d_{i-1} < d_i$, hence setting $x_k=0$ where k is determined by (5-12) also produces a left side in (5-15) less than or equal to d_i, leaving $|I_i|$-1 jobs scheduled for completion prior to their due dates, where $|I|$ is the number of elements in the set I. In addition, among all possible schedules which complete $|I_i|$ -1 jobs prior to their due-dates with probability at least equal to a, rule (5-12) specifies that schedule which

does so in minimum time. Repeated application of this rule each time an exclusion is required ensures an optimal schedule. Proof of optimality is presented in section V.B.6.

4. The Algorithm

The steps of the algorithm are

1. Order the jobs in due-date order. Set $I_0 = \phi$. Set i=1. Compute $h = (\Phi^{-1}(a))^2$. Go to step 2.
2. If i = n+1, stop. Otherwise form $I_i = I_{i-1} \cup \{i\}$. Go to step 3.
3. If $\sum_{j \in I_i} m_j + (h \sum_{j \in I_i} v_j^2)^{\frac{1}{2}} \leq d_i$, set i = i+1 and go to step 2. If not, go to step 4.
4. Select index k using the exclusion rule and remove k from I_i. Set i=i+1 and go to step 2.

Upon termination the indices remaining in set I_n represent an optimal schedule under constraints (5-7) and (5-8).

Consider the following variation. Suppose among the original set of jobs there were one or more special jobs which the decision maker wanted included in the schedule no matter what the consequences to the resulting schedule. This variation is solved optimally under the additional constraints by applying the above algorithm except that the indices of the special jobs are never included in the set I, i.e., never considered for exclusion from the schedule. In this variation it may be necessary to exclude more than one job in order to ensure that a special job be included in the schedule.

5. Example

The algorithm is illustrated with the following example. The data for the problem are shown in Table 5.1.

Job	1	2	3	4	5	6
Mean	3	7	4	8	5	9
Variance	1	3	1	4	1	1
Due Date	6	11	14	18	24	31

Table 5.1. Example problem data.

For an arbitrary choice of a = .84, h is computed to be approximately 1.0 at step 1. A new iteration is initiated each time step two is executed for the next job in the sequence. The six iterations required to solve the example are discussed below.

1. Index i=1, $I_1 = \{1\}$, $m_1 + v_1 = 4$ which is less than 6, the due-date of job 1, so job 1 satisfies (5-8) and i is set equal to 2.

2. $I_2 = \{1,2\}$, $m_1 + m_2 + (v_1^2 + v_2^2)^{\frac{1}{2}} \simeq 13.16$ which exceeds due-date 11 for job 2, so some job must be excluded. In step 4, for r=1, $m_1 + v_1 = 4$, and for r=2, $m_2 + v_2 \simeq 8.16$, so k=2 and I_2 is now set equal to {1}.

3,4,5. In iterations 3,4, and 5 no jobs require exclusion.

6. For i=6, $I_6 = \{1,3,4,5,6\}$, and

$m_1 + m_3 + m_4 + m_5 + m_6 + (v_1^2 + v_3^2 + v_4^2 + v_5^2 + v_6^2)^{\frac{1}{2}} \cong 31.83$

which exceeds due date 31 for job 6, so one more job must be excluded. At step 4, k is found to be 6 and 6 is removed from I_6. The remaining indices in I_6 represent an optimal schedule.

If variances were assumed to be zero, this algorithm corresponds exactly with Moore's and the optimal schedule would be {1,2,3,5,6}. However, due to the uncertainty of processing times and the requirement to complete the selected schedule with probability at least equal to .84, two jobs were required to be excluded instead of one.

6. Proof of Optimality

It will be assumed that application of the algorithm of section V,B,4 to the problem with constraints (5-7) and (5-8) using (5-12) as the exclusion decision rule does not produce an optimal schedule and a contradiction is shown using an inductive procedure similar to that of Ref. 39 .

Let B be a set containing k jobs excluded by the algorithm. Assume there exists another set C containing k-1 jobs which allows n-k+1 jobs to be completed prior to their due-dates, indicating that the solution obtained by the algorithm is not optimal. A contradiction to this assumption will be shown.

Let B_i be the set of indices for jobs excluded from the sequence up to and including consideration of the i^{th} job in the sequence upon appliation of the algorithm. Let

C_i be that subset of C containing job numbers which are less than or equal to i. Clearly $C_n = C$.

The proof will inductively show that

$$|C_i| \geq |B_i| \quad \text{for } i=1,\ldots,n. \tag{5-16}$$

The assumption requires for all i that

$$\sum_{j=1}^{j=i} m_j x_j + (h \sum_{j=1}^{j=i} v_j^2 x_j)^{\frac{1}{2}} \leq d_i \tag{5-17}$$

where $x_j=0$ if j is in C_i and $x_j=1$ otherwise.

Let J_{a_1} be the first job encountered which fails the test in step 3 of the algorithm. The induction begins by showing that C_{a_1} must contain one or more job index numbers. (Clearly $|B_i| \leq |C_i|$, for $i=1,\ldots,a_1-1$.)

Given index a_1, the following condition exists:

$$\sum_{j=1}^{j=a_1} m_j + (h \sum_{j=1}^{j=a_1} v_j^2)^{\frac{1}{2}} > d_{a_1} . \tag{5-18}$$

Going to step 4, job J_{b_1} is selected for exclusion from the schedule using the exclusion rule with $I_{a_1} = \{1,2,\ldots,a_1\}$ and b_1 is placed in B_{a_1}, making $|B_{a_1}| = 1$.

Inequality (5-17) requires that there be at least one c_1 in $\{1,2,\ldots,a_1\}$ such that J_{c_1} is excluded from the schedule, hence

$$|C_{a_1}| \geq 1 = |B_{a_1}| .$$

Note that if $|C_{a_1}| = 1$, the left side of (5-18) when J_{b_1} is excluded is less than or equal to the same sum when J_{c_1} is excluded.

Proceeding to the inductive step, suppose the algorithm has considered $1 < p \leq n$ jobs and that m have been excluded. It is thus assumed that

$$m = |B_p| \leq |C_p| = q \quad .$$

Two cases need be considered. If $m < q$ and/or no job requires exclusion at iteration p+1, then the induction requirement (5-16) is trivially satisfied at iteration p+1. If m=q and an exclusion is required at iteration p+1 it must be shown that $m+1 = |B_{p+1}|$ is less than or equal to $|C_{p+1}|$. Note that application of rule (5-12) each time an exclusion is required and the fact that (5-17) must be satisfied guarantees that the left side of (5-13) with i=p and $x_j=0$ for j in B_p and $x_j=1$ otherwise will be less than or equal to the same sum with $x_j=0$ for j in C_p and $x_j=1$ otherwise.

Since exclusion is required at iteration p+1,

$$\sum_{j=1}^{j=p+1} m_j x_j + \left(h \sum_{j=1}^{j=p+1} v_j^2 x_j\right)^{\frac{1}{2}} > d_{p+1} \qquad (5\text{-}19)$$

where $x_j=0$ for j in B_p, and $x_j=1$ otherwise. But the left side of (5-19) with $x_j=0$ for j in C_p and $x_j=1$ otherwise exceeds the left side of (5-19), and thus (5-17) will be

violated unless at least one more job index number is contained in C_{p+1}. Thus

$$|C_{p+1}| \geq |B_{p+1}| .$$

In particular, let p=n-1 and the desired contradiction to the assumption is achieved.

C. REMARKS

In this section, one problem formulation was solved optimally through application of the chance constrained programming technique of Charnes and Cooper [Ref. 9]. Other formulations may suggest application of different techniques in probabilistic programming. For example, for problems in which some or all of the escape times are uncertain and an integer programming formulation appears tractible except for the random elements in the constraints, it may be possible through application of "Linear Programming Under Uncertainty" techniques [Refs. 14 ,15 , 29] to move all random elements into the objective function and minimize an expected value.

The applicability of the branch and bound technique to problems with random investigation times is unclear, however, the possibility is enhanced due to the work of Clark [Ref. 11] in which a method of approximating moments for the maximum of a set of normal random variables is developed, a procedure which could be used in the development of branching rules.

VI. NON-ZERO RELEASE TIMES

A. INTRODUCTION

Chapter III considers problems for which a known ordering is externally imposed on all solutions and all targets are available at time zero. These problems are demonstrated to have efficient solution methods. Assuming that targets are all available at time zero corresponds to the standard assumption in job-shop scheduling that all jobs are available for processing at time zero. This assumption is very restrictive and seriously conflicts with the actual circumstances surrounding both job-shop and investigation problems. In both areas there can be, and usually is, plentiful information concerning future arrivals. This chapter considers the implications of using this information during optimization instead of ignoring it.

The analytical treatment of job-shop problems is very seriously complicated by the introduction of non-zero release times, (i.e., ready times or earliest start times), and as a result, very few attempts have been made at addressing the problem [Ref. 12]. Each such attempt treats an objective related to flow time. The zero release time assumption is made to allow analytical treatment and is tolerated only because of the rate at which things happen in the job-shop. Schedules can be developed for those jobs currently in the shop and used until the number of new

arrivals increases to some point where the current schedule is no longer meaningful or acceptable, at which point a new schedule is developed, taking into account all old unprocessed jobs and all new arrivals.

In investigation problems, the rate at which events occur can be much faster than in the job-shop and the result of suboptimizing over the set of objects currently in the region of interest can render the schedule obtained meaningless. If the investigator has information concerning the location and motion of objects which are currently outside the region but will at some future time enter and pass through, then it is desirable to attempt to take these objects into account when selecting a path to follow. For example, in the missile defense problem where the investigator is a missile system and objects are enemy missiles and aircraft proceeding toward the formation, virtually all decisions regarding the order in which targets will be taken under fire should be made prior to the entry of any target into the missile envelope of the missile system. Thus, in this case, it is desirable to have the analytical treatment of the problem complete before any object enters the region. Any method which assumes zero release times for all objects is thus worthless in such cases.

This chapter presents the results of analysis of the fundamental structure of the non-zero availability problem. The notation and terminology used throughout this chapter is that of job-shop scheduling as contained in Ref. 12,

because the set of assumptions made to allow the analysis casts the investigation problem into the exact form of a particularly easy to understand, (yet very difficult to solve), job-shop scheduling problem.

The theory developed is exploited in the construction of simple algorithms for obtaining optimal solutions in certain cases. For those cases where a combinatorial approach cannot be avoided, it is shown that the theory can be used to drastically reduce the feasible solution space for a problem and also provide rapid accelerations for any combinatorial algorithmic approach.

1. Problem Statement

The notation, most of which is that of Reference 12, is as follows.

J_i = job number i

[i] = a subscript to denote the i^{th} element of a sequence

$J_{[i]}$ = the i^{th} job in a schedule

p_i = processing time of J_i

r_i = release time of J_i

d_i = due-date of J_i

W_i = waiting time until the start of J_i in some schedule

C_i = completion time of J_i in some schedule

$L_i = C_i - d_i$ = lateness of J_i in some schedule

T_i = $\max(0,L_i)$ = tardiness of J_i in some schedule

$(i_1,i_2,\ldots,i_n)$ = a permutation of numbers $1,2,\ldots,n$

DD order = non-decreasing order of due-dates

DD schedule = one with jobs arranged in DD order

R order = non-decreasing order of release times

R schedule = one with jobs arranged in R order

$i \leftarrow j$, spoken "i before j", means that J_i is scheduled before J_j

Consider the job-shop problem with one machine and n jobs in which, for each i, job i is released to the shop for processing at time r_i, takes p_i units of time to process, and is due at time d_i. A job once started is processed until completion. Let set-up be sequence independent and included in the processing time for each job. Jobs which cannot be processed prior to their due-dates are not processed, (or are processed at the end of the schedule in any order). The objective is to maximize the number of jobs processed prior to their due-dates.

Letting m be the number of elements in permutation π, the objective function, $f(\pi)$, for this problem is simply m and the problem statement is: determine a π which will

maximize m

subject to

$$\sum_{j=1}^{j=i} p_{\pi_j} \leq d_{\pi_i} \qquad i = 1,\ldots,m$$

$$\sum_{j=1}^{j=i-1} p_{\pi_j} \geq r_{\Pi_i} \qquad i = 1,\ldots,m$$

The set of all possible solutions π includes all permutations $(i_1,i_2,\ldots,i_k)$ of integers $\{1,2,\ldots,n\}$ for $k = 1,\ldots,n$ in which each element i_j in $\{1,2,\ldots,n\}$ appears at most once.

2. Separability

It is possible for a problem to be separated, or partitioned, into smaller subproblems, each of which can be solved independently. Two conditions under which this partitioning is possible are presented in theorems 6.1 and 6.2

Let jobs be indexed in DD order.

Theorem 6.1: Schedule all jobs in reverse due-date order, ignoring release times and observing due-dates, such that all jobs are completed as late as possible. Let there be k occurrences of machine idle time in this schedule. The n job problem can be partitioned into k+1 independent subproblems of size $n_1,n_2,\ldots,n_{k+1}$ where $n_1 + n_2 + \ldots + n_{k+1} = n$. Let $S_1,\ldots,S_{k+1}$ be optimal solutions to subproblems $1,2,\ldots,k+1$. The optimal solution to the n job problem is $S_1\ S_2\ \ldots\ S_{k+1}$, sets being arranged in DD order.

Proof: Let Figure 6.1 represent the schedule described showing the first occurrence of idle time.

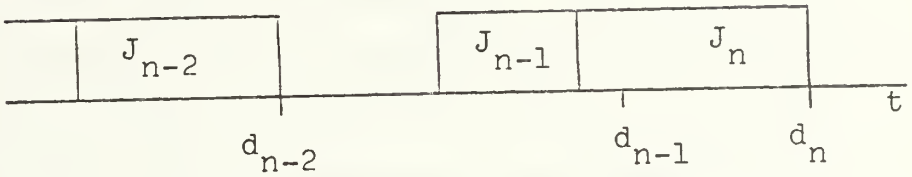


Figure 6.1

Note that whatever partial schedule is considered from among jobs {1,2,...,n-2}, it must be completed prior to d_{n-2} or it will not be feasible, i.e., it will contain at least one job whose completion time exceeds its due-date. Note also that whatever partial schedule is considered for jobs after J_{n-2}, it will never be necessary to have it start prior to d_{n-2}. Hence, if an iterative procedure is used to obtain an optimal schedule which considers jobs in DD order, the process can be terminated with J_{n-2} and restarted, considering only jobs J_{n-1} and J_n starting at time $\max [d_{n-2}, r_{n-1}]$.

The same reasoning applies to all cases of idle time. The optimal schedule to the n job problem will clearly be the union of the optimal schedules to the sub-problems with each job scheduled as early as possible.

Theorem 6.2: Schedule all jobs in R order, observing release times and disregarding due-dates, and scheduling each job as early as possible. Let there be k occurrences of machine idle time in the schedule. The n job problem can be partitioned into k+1 independent subproblems of size $n_1, n_2, \ldots, n_{k+1}$ where $n_1 + n_2 + \ldots + n_{k+1} = n$. Let $S_1, \ldots, S_{k+1}$ be optimal solutions to the subproblems $1, 2, \ldots, k+1$. The optimal solution to the n job problem is $S_1 \; S_2 \; \ldots \; S_{k+1}$, sets being arranged in R order.

Proof: Let Figure 6.2 represent a partial schedule developed as described above showing the first occurrence of idle time.

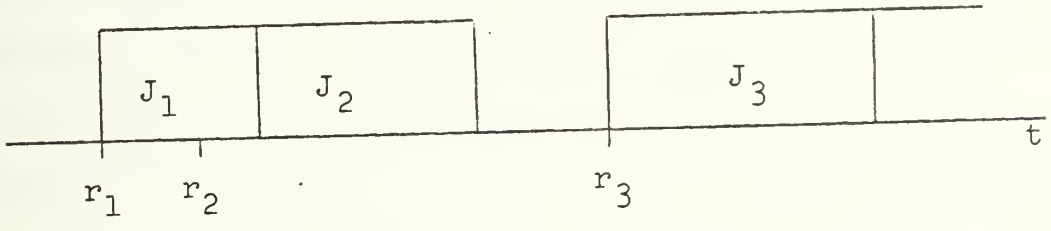


Figure 6.2

Note that whatever partial schedule is considered from among jobs $\{3, \ldots, n\}$, it can start no earlier than r_3. Note also that no matter what the due-dates are for jobs J_1 and J_2, it will never be necessary to have a completion time for any feasible partial schedule constructed through

consideration of only jobs J_1 and J_2 which exceeds r_3. Hence, an iterative procedure which builds an optimal schedule by considering jobs in R order can terminate after consideration of J_2 and restart at time r_3, ignoring all preceeding jobs.

The same reasoning applies to all occurrences of idle time. The optimal schedule for the n job problem will clearly be the union of the optimal schedules for the subproblems with all jobs scheduled as early as possible.

Lemma 6.2.1: If the DD schedule is identical to the R schedule for a problem, then the R partition can be refined using the DD partition, or vice versa.

The validity of the lemma is obvious since both partitions apply to the same sequence when DD = R. The following example is used to demonstrate the applicability of the lemma. Let P be the DD partition and Q the R partition such that

$$P = \{\{1,2,3,4\},\{5,6,7\}\}$$

$$Q = \{\{1,2\},\{3,4,5,6,7\}\}.$$

Then the refined partition is

$$\{\{1,2,\},\{3,4\},\{5,6,7\}\}.$$

In all that follows, each problem discussed is assumed to be reduced to minimum size and jobs renumbered using integers 1,2,...,n.

B. DUE-DATE SEQUENCES

Under certain circumstances it is possible to determine that there exists an optimal schedule in DD order. In these instances the search for an optimal schedule is greatly simplified in that only one of the n! possible trial permutations, that being a permutation containing all n elements {1,2,...,n}, need be considered. Letting jobs be indexed in DD order, the problem reduces to determining the minimum number of elements which must be removed from the permutation (1,2,...,n) in order to produce feasibility in the schedule represented by the remaining elements. A very simple algorithm is presented in Section VI,B,2 which obtains optimal schedules in such instances. Section VI,B,1 presents conditions under which there exists an optimal schedule in DD order developed through a series of pairwise comparisons of job parameters.

1. Partial DD Orderings

When presented with a set of jobs and their parameters, p_i, r_i, and d_i, it is possible to state, depending upon the results of pairwise comparisons of these parameters, that certain trial permutations need never be considered in any search for an optimal schedule. In some instances, all but one trial permutation can be eliminated. Theorems 6.3,

6.4, and 6.5 derive the most general conditions known under which this is true for the DD permutation.

The first theorem eliminates trial permutations from consideration through comparison of adjacent pairs of jobs in any optimal schedule.

Theorem 6.3: For every pair of jobs J_i and J_{i+1} such that $r_i \leq r_{i+1}$, if there exists an optimal schedule containing both J_i and J_{i+1} in which $i+1 \leftarrow i$, then there also exists an optimal schedule in which $i \leftarrow i+1$.

Proof: The form of the proof is to assume that S^1 is an optimal schedule with $i+1 \leftarrow i$ and show that schedule S^2, which is identical to S^1 except that the two jobs are switched such that $i \leftarrow i+1$, is feasible, and thus also optimal. Note that all jobs in S^1 scheduled before J_{i+1} and after J_i will be uneffected by the switch, (see Figure 6.3). Since S^1 is feasible, $d_i \geq W_k$ and $r_{i+1} \leq C_j$. Let

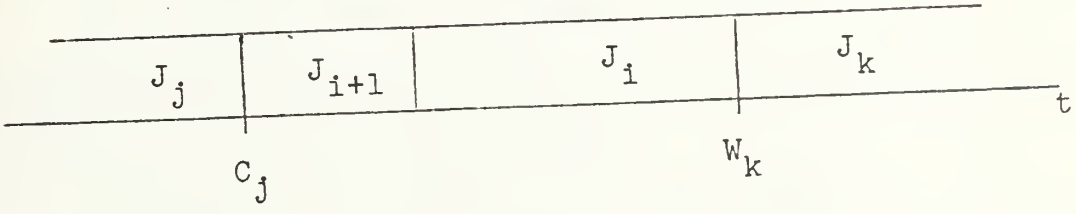


Figure 6.3

$\overline{W}_i$ be the start time of J_i after the switch and $\overline{C}_{i+1}$ be the completion time of J_{i+1} after the switch. The facts $d_i \leq d_{i+1}$ and $r_i \leq r_{i+1}$ imply that

$$r_i \leq C_j = \overline{W}_i \quad \text{and} \quad \overline{C}_{i+1} = W_k \leq d_{i+1},$$

so J_i and J_{i+1} will both be feasibly schedule in S^2, showing that S^2 is also optimal.

The second theorem eliminates trial permutations from consideration through comparison of arbitrary pairs of jobs.

Theorem 6.4: For pairs of jobs J_i and J_j with $i < j$, $r_i > r_j$, and $p_i + p_j > d_i - r_j$, there exist no optimal schedules containing both J_i and J_j with $j \leftarrow i$.

Proof: The proof will assume the existence of an optimal schedule with $j \leftarrow i$ and show a contradiction. Assume

1. $p_i + p_j > d_i - r_j$ and
2. there exists an optimal schedule with $j \leftarrow i$.

In any feasible schedule with $j \leftarrow i$, $w_j \geq r_j$ and $c_i \leq d_i$. Clearly $c_i \geq w_j + p_i + p_j$, hence

$$c_i \leq d_i \Rightarrow w_j + p_i + p_j \leq d_i, \text{ and}$$

$$r_j \leq w_j \Rightarrow r_j + p_i + p_j \leq c_i \leq d_i,$$

$$\Rightarrow p_i + p_j \leq d_i - r_j,$$

which is the desired contradiction.

Under the conditions of the theorem, it is necessary only to consider trial permutations with job pairs J_i, J_j in DD order.

The partial orderings of jobs obtained through application of theorems 6.3 and 6.4 can be represented in the form of a directed graph. Let jobs $J_1, J_2, \ldots, J_n$ be represented by nodes numbered 1,2,...,n and each precedence relation " $\leftarrow$ " developed through application of theorems 6.1 and 6.2 be represented by an arc from node j to node i if $i \leftarrow j$. This graph is called the DD graph and is illustrated with the following example. Consider the job set described in Table 1.

i	1	2	3	4	5
p_i	2	3	2	4	5
r_i	0	0	3	2	5
d_i	4	6	7	14	15

Table 1. Example Problem Data.

Application of theorem 6.3 results in the following precedence relations being established.

$1 \leftarrow 2$ $2 \leftarrow 3$ $3 \leftarrow 5$ $4 \leftarrow 5$

$1 \leftarrow 3$ $2 \leftarrow 4$

$1 \leftarrow 4$ $2 \leftarrow 5$

$1 \leftarrow 5$

Application of Theorem 6.4 establishes relation 3 ← 4.
The DD graph for this example is illustrated in Figure 6.4

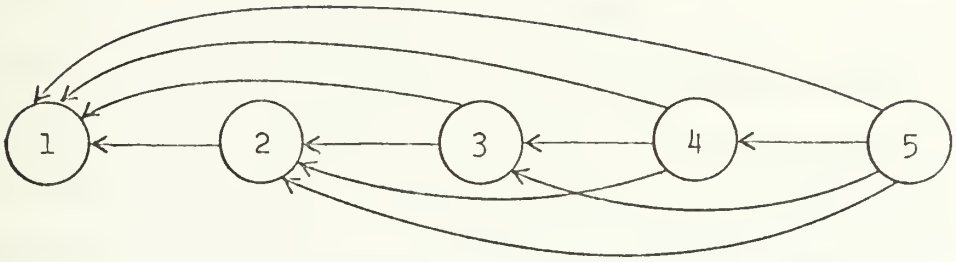


Figure 6.4. Example DD Graph.

This graph is used to present a condition under which there exists an optimal schedule in DD order.

Theorem 6.5: In the DD graph, if there exists a hamiltonian path from node n to node 1, then there exists an optimal schedule in DD order.

Proof: Theorems 6.3 and 6.4 develop only relations between job pairs which, when represented graphically, are reverse arcs. The only possible hamiltonian path constructed only of reverse arcs is the reverse DD sequence.

Under the conditions of Theorem 6.5 and due to the transitivity of the " ← " relation, the partial orderings of Theorems 6.3 and 6.4 impose a complete ordering on the set 1,2,...,n. Hence all trial permutations except (1,2,...,n) are eliminated from consideration.

For the DD graph depicted in Figure 6.4 in which only reverse arcs are allowed, the only possible hamiltonian path from node five to node one is (5,4,3,2,1) and it is trivial to determine if such a path exists. In the Sections which follow the DD graph will be augmented with a set of forward arcs, and again it will be desirable to determine if there exists one or more hamiltonian paths. The practical applicability of Theorem 6.5 might thus appear to be doubtful as there do not presently exist any efficient methods of specifying hamiltonian paths in arbitrary graphs, [Refs. 7 and 18]. However, for the graph associated with this problem, a simple technique has been developed which identifies all such paths and is presented in Section VI,D,2.

2. Due-Date Algorithm

Under the conditions of Theorem 6.5, a problem can be solved very easily through application of the following algorithm. Let

I_i = an ordered set of job indicies representing a partial schedule developed during consideration of jobs 1,2,...,i in the DD sequence.

$\bar{S}(j)$ = the reduction in completion time of a partial schedule as a result of excluding J_j.

Algorithm:

1. Index all jobs in DD order. Set $I_0 = \phi$. Set i = 1. Go to step 2.

2. If $i = n+1$, stop. Otherwise form $I_i = I_{i-1} \cup i$. Go to step 3.

3. If $\sum_{j \in I_i} p_j \leq d_i$, set $i = i+1$ and go to step 2. If not, go to step 4.

4. Compute $\bar{S}(j) = \min_{\substack{v \in I_i \\ v>j}} [p_j, \min(w_v - r_v)]$ for all $j \in I_i$. Select index k such that $\bar{S}(k) = \max_{j \in I_i} \bar{S}(j)$ and remove k from I_i. Set $i = i+1$ and go to step 2.

The proof that the DD algorithm produces an optimal schedule rests on the facts that while iteratively building a schedule in DD order, if an infeasible partial schedule is produced, only one job must be excluded to obtain feasibility, and that the rule of step 4 selects that job for exclusion which allows the remaining partial schedule to be completed in minimum time. Prior to proving the optimality of the algorithm, these two facts are stated as Theorems and proved.

Let C(I) be the completion time of schedule I.

Theorem 6.6: Given that there exists an optimal schedule in DD order, if partial schedule I_{k-1} is feasible, but for $I_k = I_{k-1} \cup \{k\}$, $C(I_k) > d_k$, then one and only one job must be excluded from I_k to produce feasibility.

Proof: Since I_{k-1} is feasible and $I_k = I_{k-1} \cup \{k\}$, k can be removed from I_k to produce a feasible schedule.

The usefulness of the Theorem is in showing that at most one job must be excluded from I_k to produce feasibility.

Theorem 6.7: Given that there exists an optimal schedule in DD order and that partial schedule I_{k-1} is feasible, but for $I_k = I_{k-1} \quad \{k\}$, $C(I_k) > d_k$, then excluding m from I_k where

$$\bar{S}(m) = \max_{i \in I_k} \bar{S}(i)$$

allows the remaining schedule $I_k - \{m\}$ to be

a) feasible

b) have minimum completion time as compared to all partial schedules $I_k - \{j\}$, $j \in I_k$.

Proof: a) feasibility;

All jobs in $I_k - \{k\}$ are feasibly scheduled in I_k and will remain so if any $j \in I_k$, $j \neq k$ is excluded, but for J_k, $C_k > d_k$. Clearly $C(I_k) = C_k$, $C_k - d_k \leq p_k$, and $\bar{S}(m) \geq p_k$. So,

$$C(I_k - \{m\}) = C(I_k) - \bar{S}(m) \leq C(I_k) - p_k = C_k - p_k,$$

thus $C(I_k - \{m\}) \leq d_k$ and $I_k - \{m\}$ is a feasible schedule.

b) minimum completion time:

$$\bar{S}(m) = \max_{j \in I_k} \bar{S}(j)$$

thus

$$C(I_k) - \bar{S}(m) \overset{<}{=} C(I_k) - \bar{S}(j) \quad \text{for all } j \text{ in } I_k$$

and

$$C(I_k - \{m\}) \overset{<}{=} C(I_k - \{j\}) \quad \text{for all } j \text{ in } I_k$$

so

$I_k - \{m\}$ has minimum completion time.

The proof that the DD algorithm produces an optimal schedule is now presented.

Assume the DD algorithm is applied to a problem and a set of jobs E is excluded. Assume also that there exists another set of jobs F with $|F| < |E|$ for which $\{1,2,\ldots,n\} - F$ is a feasible schedule, thereby indicating that solution $\{1,2,\ldots,n\} - E$ is non-optimal. The proof will show by induction that $|E| \leq |F|$, no matter what the set F is, thus contradicting the second assumption.

Let E_k be the set of all jobs removed by the DD algorithm up to and including consideration of J_k, i.e.,

$$E_k = \{e | e \in E \text{ and } e \leq k\} .$$

Similarly, let

$$F_k = \{f | f \in F \text{ and } f \leq k\} .$$

The induction begins by showing that $|E_k| \leq |F_k|$ where k is the first iteration at which a job is required to be excluded.

To the job set of the problem, apply the algorithm until at iteration k, I_{k-1} is feasible, $I_k = I_{k-1} \cup \{k\}$ is formed and $C(I_k) > d_k$. By Theorem 6.6, one job from I_k must be removed to produce feasibility. Index m is deleted from I_k where for job J_m,

$$\overline{S}(m) = \max_{j \in I_k} \overline{S}(j).$$

Hence $|E_k| = 1$.

But since $C(I_k) > d_k$, there must be at least one f in F such that $1 \leq f \leq k$, or else partial schedule $\{1,2,\ldots,k\} - F_k$ will not be feasible. Hence

$$|F_k| \geq |E_k| = 1.$$

Note that if $|F_k| = |E_k| = 1$, then by Theorem 6.7,

$$C(\{1,2,\ldots,k\} - F_k) \geq C(\{1,2,\ldots,k\} - E_k) .$$

Now suppose at iteration t-1, I_{t-1} is feasible and at iteration t, $I_t = I_{t-1} \cup \{t\}$ is formed and $C(I_t) > d_t$. Let

$$q = |E_{t-1}| \leq |F_{t-1}| = q' \quad \text{and}$$

$$C(\{1,2,\ldots,t-1\} - E_{t-1}) \leq C(\{1,2,\ldots,t-1\} - F_{t-1}).$$

Two cases need be considered.

1. If $q < q'$, increasing the order of E by one will not disturb the inequality $|E_t| \leq |F_t|$.
2. If $q = q'$, note that

$$C(I_t) = C(\{1,2,\ldots,t\} - E_{t-1})$$
$$= C(\{1,2,\ldots,t-1\} - E_{t-1}) + p_t > d_t,$$

but

$$d_t < C(\{1,\ldots,t-1\} - E_{t-1}) + p_t$$
$$\leq C(\{1,\ldots,t-1\} - F_{t-1}) + p_t,$$

so there must exist at least one f in F such that f is not in F_{t-1} and $1 \leq f \leq t$, or else the partial schedule $\{1,\ldots,t\} - F_t$ is infeasible. Thus

$$|F_t| \geq |F_{t-1}| + 1 = q + 1,$$

which implies that

$$q' + 1 \geq |F_t| \geq |E_t| = q + 1.$$

With this, the induction is complete and the contradiction to the assumption that $|F| < |E|$ is evident.

The DD algorithm provides optimal schedules for problems with due-dates and non-zero release times and is thus a direct extension of the work of Jackson and Moore, [40], [32]. For the problem with $r_i = 0$ for $i = 1,\ldots,n$, Jackson's lemma reduces the problem to consideration of only the DD trial permutation, and Moore's algorithm efficiently extracts the optimal subpermutation. It is interesting to note that the algorithm of this section reduces identically to Moore's if $r_i = 0$, for $i = 1,\ldots,n$. This can be seen by noting that if $r_i = 0$ for all i, then $\overline{S}(j) = p_j$ for all j and the exclusion rule of step 4 reduces to selecting for exclusion from the infeasible partial schedule at the current iteration that job with the longest processing time, i.e., $p_m = \max_{j \in I_i} p_j$, which is the identical rule used by Moore.

C. RELEASE-TIME SEQUENCES

The value of this section lies in the fact that there exists a theory relevant to R sequences which parallels

that for DD sequences. As in the case of DD sequences, it is sometimes possible to determine that there exists an optimal schedule in R order. The theorems giving the conditions under which this is have nearly identical statement and proof as those of Section VI,B except that jobs are indexed differently. When there exists an optimal sequence in R order, again it is necessary only to consider a single trial permutation from among the n! possible. Section VI,C,1 presents conditions under which there exists an optimal schedule in R order. In Section VI,C,2 a simple algorithm is presented which extracts the optimal subpermutation from the R trial permutation.

1. Partial R Orderings

As shown in Section VI,B,1, pairwise comparisons of job parameters can be employed to eliminate from consideration certain trial permutations. Theorems 6.8, 6.9, and 6.10 present conditions under which all but the R trial permutation can be eliminated from consideration.

Let jobs be indexed in R order. Theorem 6.8, as does 6.3, eliminates trial permutations from consideration through comparison of adjacent jobs in any optimal schedule.

Theorem 6.8: For every pair of jobs J_i and J_{i+1} with $d_i \leq d_{i+1}$, if there exists an optimal schedule in which $i+1 \leftarrow i$, then there exists also an optimal schedule in which $i \leftarrow i+1$.

The proof in this case is identical in statement to the proof of Theorem 6.3, given the revised indexing procedure, and thus will be omitted.

Under the conditions of Theorem 6.8, it is unnecessary to consider any trial permutation in which i+1 preceeds i.

Lemma 6.8.1: If the DD trial permutation is identical to the R trial permutation, there exists an optimal schedule in R order.

Proof: In this case, Theorem 6.8 imposes a complete ordering on the job set, i.e., there exists a hamiltonian path from node n to node 1 in the graphical representation, and by Theorem 6.5, there exists an optimal path in DD order, which is the same as the R order.

It is interesting to note that Lemma 6.8.1 gives a condition under which there exists an optimal schedule in both R and DD order which is independent of job processin times.

The following theorem is similar to Theorem 6.4 in that it eliminates trial permutations through comparison of job parameters for arbitrary pairs of jobs.

Theorem 6.9: For all jobs J_i and J_j with $i < j$, $d_i > d_j$, and $p_i + p_j > d_i - r_j$, there exists no optimal schedule containing both J_i and J_j with $j \leftarrow i$.

Once again the statement of the proof is identical to that for Theorem 6.4, given the new indexing procedure, and thus will be omitted. The usefulness of this theorem is that when considering trial permutations in search for one which contains an optimal subpermutation, it is only necessary to consider those in which $i \leftarrow j$.

Theorems 6.4 and 6.9 are nearly identical in statement and proof and differ mainly in the method of indexing used. In order to illustrate the difference, a graphical comparison is made. In Figure 6.5 jobs are indexed in R order and $d_i - r_j$ of Theorem 6.9 is shown. In Figure 6.6

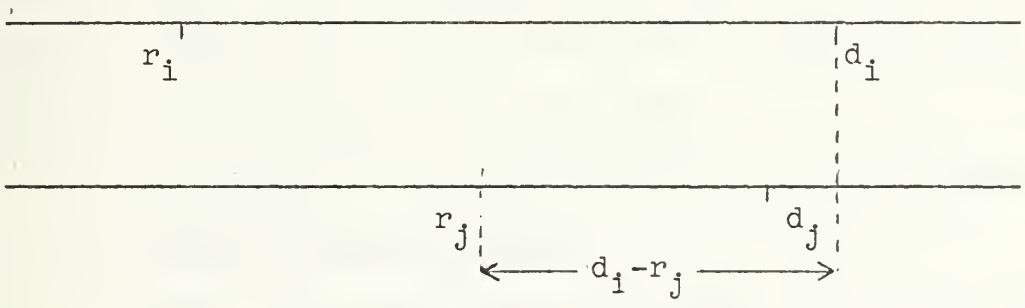


Figure 6.5. Illustration of Theorem 6.9

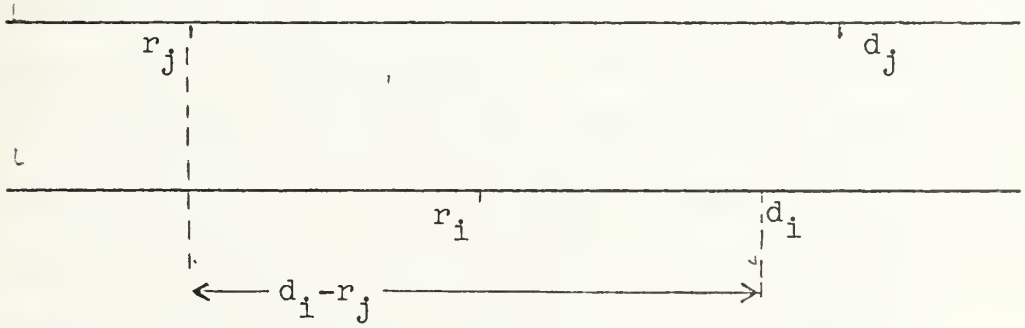


Figure 6.6. Illustration of Theorem 6.4

jobs are indexed in DD order and $d_i - r_j$ of Theorem 6.4 is illustrated. These quantities are clearly different and only accidentally bear the same name due to the method of indexing used.

As in the DD case, it is possible to represent the partial orderings developed through application of Theorems 6.8 and 6.9 graphically. Let nodes be numbered in R order and let arcs be defined as in Section VI,B,1. This graphical representation can be used to state a condition under which there exists an optimal schedule in R order.

Theorem 6.10: In the R graphical representation, if there exists a hamiltonian path from node n to node 1, then there exists an optimal schedule in R order.

The proof is identical to the proof of Theorem 6.5.

2. Release-Time Algorithm.

Under the conditions of Theorem 6.10, it is only necessary to consider the R trial permutation in the search for an optimal schedule. The following algorithm, which exploits this fact, is an inverted form of the due-date algorithm of Section VI,B,2. Let I^i be an ordered set of job indices representing a partial schedule developed by considering jobs i, i+1,...,n in R order. Let W(I) be the start time of set I, and $\hat{S}(j)$ be the increase in start time of the partial schedule if J_j is excluded.

1. Index all jobs in R order. Set $I^{n+1} = \phi$.
   Set i = n. Go to step 2.

2. If $i = 0$, stop. Otherwise form $I^i = I^{i+1} \bigcup \{i\}$. Go to step 3.

3. Compute $W(I^i) = \min[W(I^{i+1}) - p_i, d_i - p_i]$. If $W(I^i) > r_i$, set $i = i-1$ and go to step 2. Otherwise, go to step 4.

4. Compute $\hat{S}(j) = \min[p_j, \min_{i \leq v < j} (d_v - C_v)]$ for all j in I^i. Select index m such that $\hat{S}(m) = \max_{j \in I^i} \hat{S}(j)$ and remove m from I^i. Set $W(I^i) = W(I^i) + p_m$. Set $i = i-1$ and go to step 2.

The proof that the algorithm will produce an optimal schedule is simplified by first establishing two facts; first that while iteratively building partial schedules, if an infeasible partial schedule is developed, at most one job need be excluded to produce feasibility, and secon that the exclusion rule of step 4 excludes that job which allows the remaining partial schedule to have the latest possible completion time. These facts are stated as theorems and proved.

Theorem 6.11: Given that there exists an optimal schedule in R order, if partial schedule I^{k+1} is feasible and $I^k = I^{k+1} \bigcup \{k\}$ is not feasible because $w_k < r_k$, then one and only one job must be removed from I^k to produce feasibility.

Proof: Since I^k is infeasible, at least one job must be removed to produce feasibility. Set I^{k+1} is feasible and $I^k = I^{k+1} \bigcup \{k\}$, so k can be removed from I^k to produce a feasible schedule.

Theorem 6.12: Given that there exists an optimal schedule in R order and that partial schedule I^{k+1} is feasible but $I^k = I^{k+1} \bigcup \{k\}$ is not feasible because $W_k < r_k$, i.e., $W(I^k) < r_k$, then excluding m from I^k where

$$\hat{S}(m) = \max_{j \in I^k} \hat{S}(j)$$

allows the remaining schedule $I^k - \{m\}$ to be

a) feasible

b) have latest start time as compared to all partial schedules $I^k - \{j\}$ for all j in I^k.

Proof: a) feasibility.

All jobs in I^{k+1} are feasibly scheduled in I^k and will remain so if any job j in I^k, $j \neq k$, is excluded, but for J_k, $W_k < r_k$. Clearly $W(I^k) = W_k$ and $r_k - W_k \leq p_k$, and $\hat{S}(m) \geq p_k$.

$$W(I^k - \{m\}) = W(I^k) + \hat{S}(m) \geq W(I^k) + p_k = W_k + p_k,$$

but

$$r_k \leq W_k + p_k = W(I^k - \{m\}),$$

so schedule $I^k - \{m\}$ is feasible.

b) latest start time.

$$\hat{S}(m) = \max_{j \in I^k} \hat{S}(j)$$

thus

$$W(I^k) + IC(m) \geq W(I^k) + IC(j) \quad \text{for all } j \text{ in } I^k$$

and

$$W(I^k - \{m\}) \geq W(I^k - \{j\}) \quad \text{for all } j \text{ in } I^k$$

so

$I^k - \{m\}$ has the latest possible start time.

The proof that the algorithm produces an optimal schedule is now presented. Assume the R algorithm is applied to a problem and a set of jobs E is excluded. Assume also that there exists another set of jobs F with $|F| < |E|$ for which $\{1,2,\ldots,n\} - F$ is a feasible schedule, thereby demonstrating $\{1,2,\ldots,n\} - E$ to be non-optimal. The proof will show by induction that $|F| \geq |E|$, thereby contradicting the second assumption.

Let E_k be the set of all jobs excluded by the algorithm down to and including consideration of J_k in reverse R order, i.e.,

$$E_k = \{e | e \in E \text{ and } e \geq k\}.$$

Similarly, let

$$F_k = \{f | f \in F \text{ and } f \geq k\}.$$

The induction begins by showing that $|E_k| \geq |F_k|$ where k is the first iteration where some job must be excluded from a partial schedule.

Apply the R algorithm to the job set for the problem until for some k, I^{k+1} is feasible, $I^k = I^{k+1} \cup \{k\}$ is formed and $W(I^k) < r_k$. By Theorem 6.11, only one job must be excluded to produce feasibility. At step 4 J_m is selected for exclusion where

$$\hat{S}(m) = \max_{j \in I^k} \hat{S}(j),$$

and thus $|E_k| = 1$. Note that since $W(I^k) > r_k$, there must be at least one f in F such that $k \leq f \leq n$, or else $\{k,\ldots,n\} - F_k$ will not be feasible. Hence

$$|F_k| \geq 1 = |E_k|.$$

Note that, by Theorem 6.12, if $|F_k| = |E_k|$, then

$$W(\{k,\ldots,n\} - E_k) \geq W(\{k,\ldots,n\} - F_k).$$

Now suppose at iteration $n - t$, I^{t+1} is feasible and at iteration $n - t + 1$, $I^t = I^{t+1} \cup \{t\}$ is formed and $W(I^t) < r_t$. Let

$$q = |E_{t+1}| \leq |F_{t+1}| = q' \text{ and}$$

$$W(\{t+1,\ldots,n\} - E_{t+1}) = W(\{t+1,\ldots,n\} - F_{t+1}).$$

Two cases need be considered.

1. If $q < q'$, then placing one more job index in E will not disturb the inequality $|E_t| \leq |F_t|$.
2. If $q = q'$, note that

$$\begin{aligned} W(I^t) &= W(\{t,\ldots,n\} - E_{t+1}) \\ &= W(\{t+1,\ldots,n\} - E_{t+1}) - p_t < r_t, \end{aligned}$$

but

$$\begin{aligned} &W(\{t+1,\ldots,n\} - F_{t+1}) - p_t \\ &\quad \leq W(\{t+1,\ldots,n\} - E_{t+1}) - p_t < r_t, \end{aligned}$$

so there must be at least one f in F such that $t \leq f \leq n$ and f is not in F_{t+1} or else partial schedule $\{t,\ldots,n\} - F_t$

will not be feasible. Thus

$$|F_t| \geq |F_{t+1}| + 1 = q' + 1,$$

which implies that

$$q + 1 = |E_t| \leq q' + 1 \leq |F_t|.$$

The induction is now complete and the contradiction to the assumption that $|F| < |E|$ is apparent.

D. GENERAL SEQUENCES

Sections B and C present two conditions under which all trial permutations except one can be eliminated from consideration. In Section VI,D,1 it is shown that the preceeding theory can be integrated to obtain, in certain instances, trial permutations which are known to contain optimal subpermutations which are in neither R nor DD order. Section VI,D,2 discusses the case where it is not possible to reduce the set of candidate trial permutations to a single permutation.

1. Special Cases.

Given a job set, if neither the DD nor R graphical representation contains a hamiltonian path from node n to node 1, it still may be possible to reduce the problem to consideration of a single trial permutation. Theorems 6.4 and 6.9 develop precedence relations between arbitrary pairs

of jobs which are independent of method of indexing used. Thus, the set of all relations developed by these two theorems can be coupled with either those developed by Theorem 6.3 or 6.8 to form a consistent and valid set of relations on the problem.

Let the ADD graph be the DD graph augmented with all arcs developed through application of Theorem 6.9 and the AR graph be the R graph augmented with all arcs developed through application of Theorem 6.4.

Theorem 6.13: If in either the ADD or the AR graph there exists a hamiltonian path, then there exists an optimal schedule whose order is specified by the reverse order of the hamiltonian path.

Proof: The arcs in the hamiltonian path represent a set of n-1 precedence relations which impose a complete ordering on the job set and thus all trial permutations other than the hamiltonian path, in reverse order, are eliminated from consideration.

As the starting and ending nodes of such paths are not necessarily nodes 1 and n, visualizing such paths is simplified by augmenting the graph with dummy source node s and sink node t and with arcs (i,s) for all nodes i with $\delta^+(i) = 0$ and (t,j) for all nodes j such that $\delta^+(j) = \max \delta^+(k)$, $k = 1,\ldots,n$, where $\delta^+(i)$ is the positive

degree of node i and $\delta^-(i)$ is the negative. A hamiltonian path in the original graph is a hamiltonian path from node t to node s in the new graph.

If the set of candidate trial permutations is reduced to a single one, the problem reduces to excluding the minimum number of job indices from that permutation required to produce feasibility. This optimal subpermutation is obtained by indexing all jobs in order 1,2,...,n where $(i_1,i_2,\ldots,i_n)$ is the sole candidate trial permutation and applying either the DD or the R algorithm.

2. Combinatorial Accelerations.

Let Z be the set of all trial permutations which are not eliminated by one of the theorems of Sections II or III. If there does not exist a hamiltonian path in either the ADD or the AR graph, then $|Z| > 1$ and each element of Z could contain the optimal schedule and therefore must be explicitly or implicitly investigated in any search for an optimal schedule. The combinatorial nature of such problems suggests that combinatorial programming techniques be applied. In combinatorial approaches such as implicit enumeration, branch and bound, and dynamic programming, the set of feasible solutions is implicitly enumerated in an orderly fashion by constructing partial schedules and attaching a value to each partial schedule consistent with the problem objective. Any such technique can be greatly accelerated by utilizing the information gained in the development of the ADD and AR graphs as demonstrated in the following sections.

a. Job Ranges.

Given trial permutations $\pi = (\pi_1, \pi_2, \ldots, \pi_n) \in Z$, let $\pi(j)$ be the set of all positions held by j, i.e.,

$$\pi(j) = \{i \mid \pi_i = j,\ \pi_i \in \pi \in Z\}.$$

Let

$$\pi_e(j) = \min_{i \in \pi(j)} i \quad \text{and} \quad \pi_\ell(j) = \max_{i \in \pi(j)} i .$$

Let $(\pi_e(j), \pi_\ell(j))$ be called the range of J_j in Z. Clearly any trial permutation in which j is not in some position i such that $\pi_e(j) \leq i \leq \pi_\ell(j)$ need not be considered in any search for an optimal sequence. These range boundaries are immediately available from the graphical representation. For each node j of the graph, let $\pi_e(j) = \delta^+(j)+1$ and $\pi_\ell(j) = n - \delta^-(j)$. This is seen through consideration of the example graph in Figure 6.7.

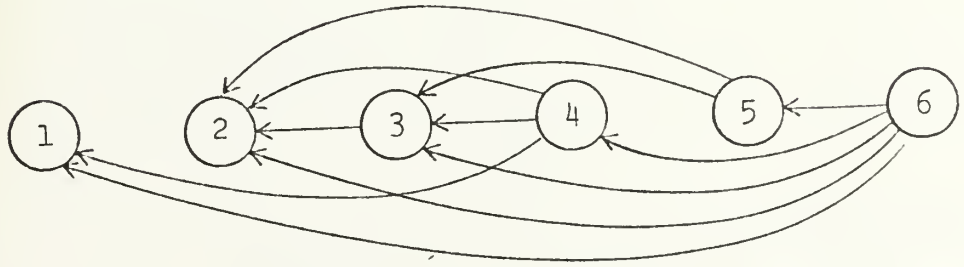


Figure 6.7 Example Graph.

Note that there are no jobs which must preceed either J_1 or J_2 because there are no arcs which are positively incident with node 1 or node 2. Thus $\pi_e(1) = \delta^+(1) + 1 = 1$, and similarly $\pi_e(2) = 1$. Note also that J_1 must preceed J_4 and J_6 in all elements of Z, hence 1 can hold no position greater than $\pi_\ell(1) = n - \delta^-(1) = 4$. Similarly, $\pi_\ell(2) = n - \delta^-(2) = 2$. Thus the range for J_1 is (1,4) and for J_2 is (1,2). The ranges for jobs J_3, J_4, J_5 and J_6 can be computed to be (2,3), (4,5), (3,5), and (5,6) respectively.

Combinatorial approaches can be accelerated by exploiting the job ranges developed to reduce the number of partial and complete schedules which need be evaluated. For the example graph, Table 6.2 is easily constructed from the job ranges computed and lists all jobs which could occupy each position in a trial permutation contained in Z

position i =	1	2	3	4	5	6
candidate	1	1	1	1	4	6
jobs for	2	2	3	4	5	
position i		3	5	5	6	

Table 6.2

Table 6.2 provides any algorithm which builds and extends partial schedules with a greatly reduced set of jobs which are eligible for use in extending whatever

current partial schedule is being considered. In the example, for instance, if a partial schedule of length two is being considered, then at most two jobs can be used to extend the current schedule, and perhaps just one. Without the table, as many as four need be investigated.

b. Complete Specification of Z.

It is possible to use the information contained in the table of the preceding section to list each trial permutation contained in Z. When |Z| is small, then total enumeration over Z using an efficient algorithm such as that of Section II may compare favorably with any combinatorial method available.

All elements of Z can be illustrated graphically by constructing a branching tree with n+1 levels as follows. Consider Figure 6.8 which is the tree constructed from Table 6.2. At level zero, create node zero. From node zero branch to level 1 to new node j where j is listed in column 1 of the table. From each node at level 1, branch to level 2 to new node k such that k is in column 2 in the table and k is not in the current path. This process is continued until level n is complete. The elements of Z are those branches containing n+1 nodes. Note that every branch in the tree is a subpermutation of some element of Z.

There exists an alternate method of depicting each element of the set Z as a hamiltonian path from a

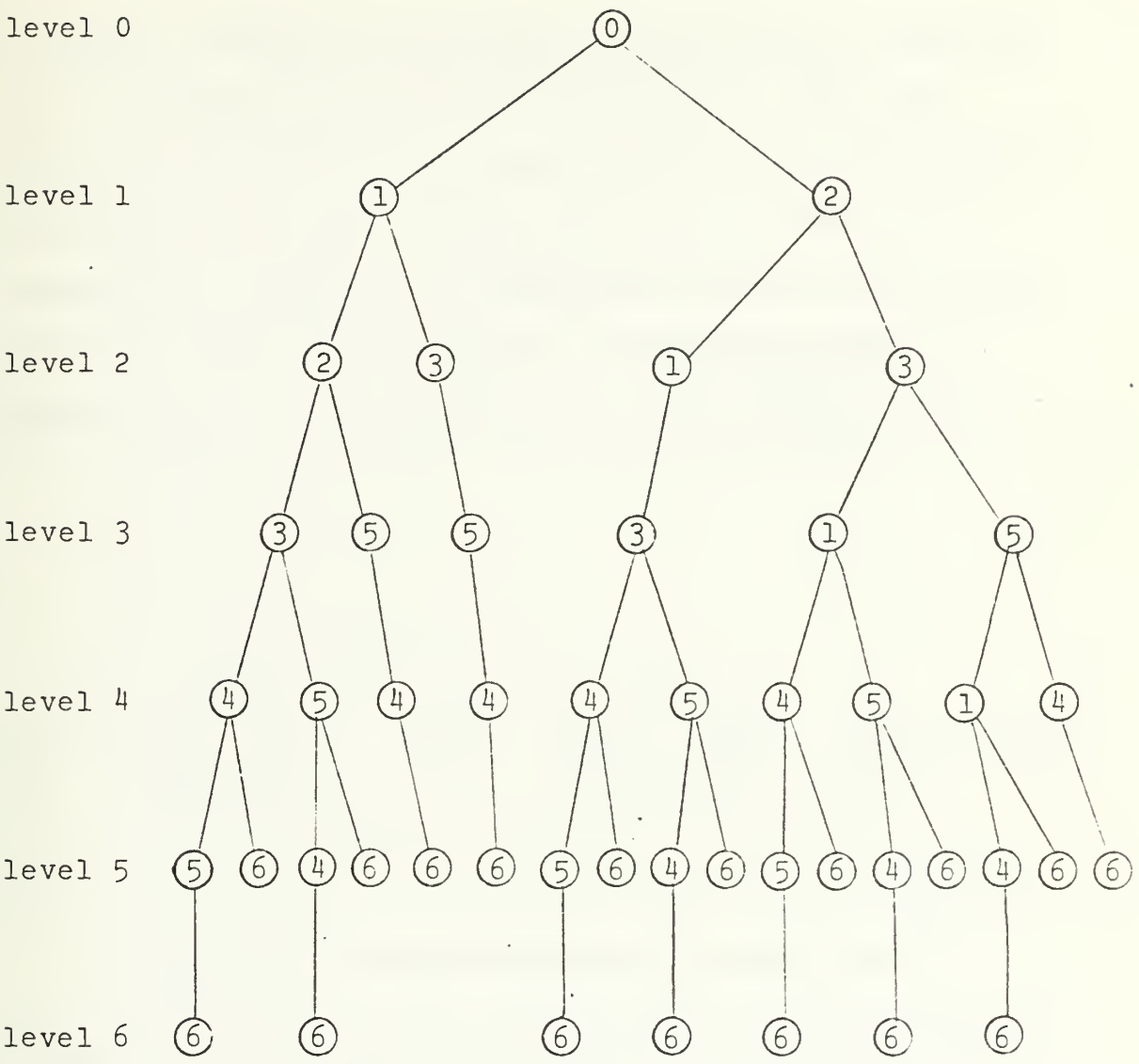


Figure 6.8

dummy source node s to a dummy sink node t in a graph containing nodes 1,2,...,n constructed as follows. Include all arcs (j,i) with i < j included in the complement to the DD graph and all arcs (i,j) with i < j, except arcs (i,k), i < k, such that there exists a path (k,j,i) in the

DD graph. Include an arc (s,i) for each node i such that $\delta^+(i) = 0$ and an arc (j,t) for each node j such that $\delta^-(j) = 0$. The resulting graph contains all elements of Z as hamiltonian paths from s to t, and, in fact, the elements in Z are the only hamiltonian paths in the graph. Figure 6.9 shows this graph for the DD graph given in Figure 6.7. This approach to the determination of all

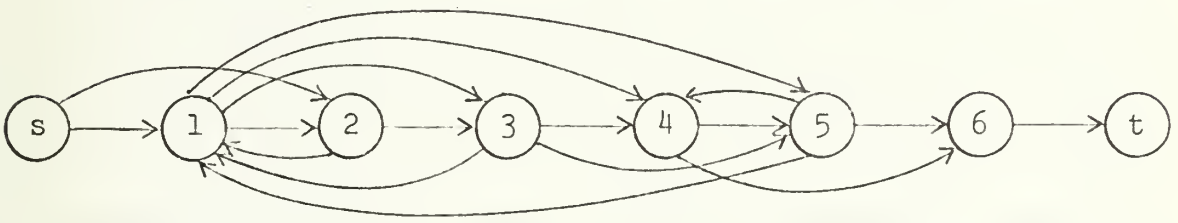


Figure 6.9 Graph Containing Elements of Z

elements of Z does not at present appear to be practical due to the lack of efficient methods for specifying hamiltonian paths in arbitrary graphs and also due to the ease with which these elements are determined through construction of the tree shown in Figure 6.8.

The tree illustrated in Figure 6.8 displays each trial permutation which could contain an optimal solution, and therefore could be used to determine all alternate optima.

If only one optimal solution is desired, the entire tree need not be constructed. Application of the methodology of branch and bound can greatly reduce the number of nodes which must be created in the tree by computing a bound for each newly created node. This bound represents a lower bound on the number of jobs which must be excluded if jobs must be processed in the order specified by order in which node labels appear in the branch. For example, consider the partial branch (0,1,3) in Figure 6.8, representing all permutations starting with elements 1 and 3. Through consideration of Table 6.2 only, a bound of one can be associated with the node labelled 3 in this branch since J_2 is not included and it is known that index 2 cannot hold any position in a trial permutation greater than two. In general, bounds can be computed on each newly created node by indexing jobs according to the order in which they appear in the associated branch and applying the DD algorithm. Obviously it is best to pursue that branch currently having the minimum bound.

3. Remarks

This section provides a theoretical basis for the development of efficient solution methods for problems with non-zero availabilities and, where possible, presents efficient algorithms. This completes the presentation of research results pertaining to the problem stated in Section II(B). In the following chapter, generalizations of the problem are considered and their applicability noted.

Avenues for future research on related problems which appear to be particularly promising are suggested.

VII. GENERALIZATIONS

A. INTRODUCTION

This section contains a discussion of several problems which are not central to the main theme of the research reported here but represent promising avenues for future research in Investigation Theory. In each case, the problem of Section II,B is directly extended and the results obtained for that problem which are applicable to the extension are summarized, along with additional facts which are immediately apparent upon consideration of special problem structure. In some cases these results suggest solution methods which are noted.

B. THE VALUE PROBLEM

The objective "minimize the number of targets to esca e uninvestigated" implies that all targets are of equal wor h or value. In many formulations this is appropriate, but n others it is not. For example, in the Market-time problem discussed in Section I, a large ship appears to be of greater value to the investigator than a small one in that it can carry a greater amount of supplies.

A direct generalization of problem (2-1) is obtained through changing the objective to maximization of the value of targets investigated. For problems whose solutions are known to have a specified ordering, this generalization has been shown to be solvable in Section III,B,6 using dynamic

programming. For other problems the general algorithm of Section IV,C,3 can be modified to handle this objective by letting the bound associated with each node in the tree be the sum of the values of targets lost rather than the number lost.

It is felt that there is sufficient structure in this extension to allow development of solution methods even more efficient than dynamic programming. It is also felt that the search for such techniques should begin through consideration of the following job-shop formulation which is called the value problem.

1. Problem Statement

Consider the job-shop scheduling problem with one machine and n jobs, each with processing time p_i, due-date d_i, and value $V_i > 0$. If job i is processed prior to its due-date, full value is realized. If not, zero value is realized. Determine the schedule with maximum value.

2. Summary of Results Applicable to the Value Problem

Jackson's lemma [Ref. 40] applies to this problem and therefore only solutions in non-decreasing order of due-dates need be considered. This is verified by noting that given any optimal solution not in due-date order, the same jobs can be rearranged in due-date order producing a feasible schedule containing the same jobs. In the following discussion, jobs are assumed to be indexed in due-date order.

As a result of the lemma, the following integer programming formulation of the problem is possible. Let

$x_i = 1$ if job i is contained in the schedule, and $x_i = 0$ otherwise. The problem is to determine x_i, $i = 1,\ldots,n$ in order to

$$\text{maximize} \quad \sum_{i=1}^{i=n} V_i x_i$$

$$\text{subject to} \quad \sum_{j=1}^{j=i} p_j x_j \leq d_i \qquad (7\text{-}1)$$

$$x_j = 0,1 \quad j = 1,\ldots,n$$

This formulation suggests that Moore's approach [Ref. 32] may be appropriate. In such an approach excluded jobs are not considered for reentry into the schedule. Assigning different values to different jobs makes such a method incorrect in that jobs once excluded by such a procedure need be considered for reentry when further infeasibilities are encountered.

The lemma reduces the problem to one of excluding jobs from the schedule (1,2,...,n) in order to maximize the value of the remaining jobs. The number of subpermutations of (1,2,...,n) which need be considered can be reduced by taking advantage of the following observations.

a. Let T_i be the tardiness of job i in some schedule. Let $T_m = \max_j T_j$ in schedule (1,2,...,n). All jobs J_i, $m < i \leq n$ are contained in all optimal solutions. This fact is seen by noting that once the tardiness in the

schedule up to and including job J_m is rectified, all subsequent jobs can be feasibly scheduled.

b. Let $1 \leq i < j \leq n$. If $V_i < V_j$ and $p_i > p_j$, then there exists no optimal schedule including J_i and excluding J_j. This fact is verified by assuming π^o is an optimal schedule containing J_i and not J_j and showing that schedule π^1 which corresponds to π^o except that it contains J_j and not J_i is feasible and has greater value. This fact eliminates from consideration all subpermutations containing i and not j. Obviously, if there exists a job J_m such that

$$V_m \geq \max_i V_i \quad \text{and} \quad p_m \leq \min_i p_i,$$

then J_m is contained in all optimal schedules.

c. Similarly, if $T_i = 0$ for $i = 1,\ldots,k-1$ and $T_k > 0$ and there exists a job J_m with $1 \leq m \leq k$ such that

$$V_m < \min_{\substack{i=1,\ldots,k \\ i\neq m}} V_i \quad \text{and} \quad p_m > \max_{\substack{i=1,\ldots,k \\ i\neq m}} p_i,$$

then there exists no optimal schedule containing J_m.

d. If there exists a job J_m, $1 \leq m \leq n$ such that

$$V_m \geq \sum_{\substack{i=1 \\ i\neq m}}^{i=n} V_i$$

then since $d_m > p_m$ there exists an optimal schedule containing J_m. This fact can be extended as follows. Let I be a set

of jobs known to be included in all optimal schedules. If there exists a job J_u such that $1 \leq u \leq n$, $u \notin I$ and

$$V_u > \sum_{\substack{i \notin I \\ i \neq u}} V_i$$

and $I \cup \{u\}$ in due-date order represents a feasible schedule, then J_u is contained in all optimal solutions.

3. Possible Solution Methods

Under certain circumstances, obtaining optimal solutions is trivial. For example, if the due-date order corresponds to a non-increasing processing time order and to a non-decreasing value order, then excluding jobs from the schedule $(1,2,\ldots,n)$ starting with J_1, then J_2, etc., until feasibility is attained will produce an optimal schedule. This is because such exclusions take the job scheduled with the largest processing time and smallest value and remove it from a position which reduces the completion time of all remaining jobs.

Another trivial case is seen by letting T^j be the maximum tardiness in the schedule $(1,\ldots,n)$ up to and including J_j. For J_1, T_1 is known to be zero or else J_1 is initially removed from the problem, hence $T^1 = 0$ also. Let i_1 be such that

$$i_1 = \min \{j | T^j > 0\},$$

and

$$i_2 = \min \{j | T^j > 0,\ j \neq i_1\},$$

etc. The sequence $(i_1, i_2, \ldots, i_k)$ represents the places in the schedule $(1, \ldots, n)$ where maximum tardiness increases. Then, if

$$T^{i_1} = \max T^i \quad i = 1, \ldots, n,$$

then the optimal schedule contains job indices $\{i_1+1, i_1+2, \ldots, n\}$ and also all jobs indices j for which y_j equals one in the following knapsack problem.

$$\text{maximize} \quad \sum_{j=1}^{i_1} V_j y_j$$

$$\text{subject to} \quad \sum_{j=1}^{i_1} p_j y_j \leq d_{i_1} \qquad (7\text{-}2$$

$$y_j = 0,1 \quad j = 1, \ldots, i_1$$

If the facts of Section VII,B,2 significantly reduce the order of the set of feasible solutions, then an implicit enumeration over the reduced set appears to be an approach which would compare favorably with the dynamic programming scheme of Section III. Further reductions in the order of the feasible solution set may be possible. This is an area suggested for future research.

C. PROBLEMS WITH MORE THAN ONE INVESTIGATOR

This dissertation deals mainly with investigation problems in which there is a single investigator. Generalization to the case of multiple investigators was shown to be possible in Section III when a known ordering is externally imposed on all optimal solutions. The solution method described is dynamic programming in which the state variable X is a vector with dimension md+1 where m is the number of investigators and d is the dimension of the region. Since this approach is computationally impractical for even small m and d, the search for practical methods seems to be a worthwhile avenue for future research.

Although the treatment of the one-machine job-shop scheduling problem is quite broad, generalizations to more than one machine are rare. One of the only significant achievements is that of Johnson and Jackson [Refs. 24 and 23] for the two machine problem in which the objective is minimization of maximum flow time.

Analysis of multi-investigator problems should clearly begin through consideration of problems with two investigators. It is felt that the following scheduling problem is an appropriate point of departure.

Consider the n job, two machine problem in which jobs have processing times p_i and due-dates d_i for $i = 1,\ldots,n$ and the objective is to maximize the number of jobs processed prior to their due-dates.

1. Integer Programming Formulation

The problem has the following integer programming formulation. Let $x_{Aj} = 1$ if J_j is scheduled on machine A, and $x_{Aj} = 0$ otherwise. Similarly for x_{Bj}. The problem is to determine x_{Aj}, x_{Bj}, $j = 1,\ldots, n$ to

$$\text{maximize} \quad \sum_{i=A,B} \sum_{j=1}^{n} x_{ij}$$

$$\text{subject to} \quad \sum_{j=1}^{k} p_j x_{Aj} \leq d_k \quad k = 1,\ldots,n \qquad (7\text{-}3)$$

$$\sum_{j=1}^{k} p_j x_{Bj} \leq d_k \quad k = 1,\ldots,n$$

$$x_{Aj} + x_{Bj} \leq 1 \qquad j = 1,\ldots,n$$

$$x_{ij} = 0,1 \quad i = A,B \quad j = 1,\ldots,n.$$

2. Discussion

Although the following is not intended to represent a complete analysis of this problem, several facts appear worthy of mention.

Let π_A and π_B represent schedules for machines A and B. There exists an optimal schedule $\pi^o = (\pi_A^o, \pi_B^o)$, that being one in which $|\pi_A| + |\pi_B|$ is maximized, for which jobs in both π_A^o and π_B^o are arranged in non-decreasing order of due-dates. This is an obvious extension of Jackson's lemma for the one machine problem. This fact indicates that an approach similar to that used by Moore [Ref. 32] for the one machine problem may be appropriate.

Such an approach would iteratively schedule jobs on either machine until a job is encountered which cannot be feasibly scheduled. When this occurs in the one machine problem, one job must be excluded. In the two machine problem this is not necessarily true. It may be possible to rearrange the schedules on A and B such that all jobs are completed prior to their due-dates and thus no job need be excluded. Each time this is not possible, excluding that job currently scheduled on either A or B with the longest processing time will produce an optimal schedule.

The difficulty in this approach is associated with the method of rearranging jobs to see if all can fit. Let J_k be the first job which cannot be scheduled feasibly on either machine and let $\pi^k = (\pi_A^k, \pi_B^k)$ be the partial schedule developed up to and including consideration of J_k. Rearranging the partial schedule $\pi^{k-1} = (\pi_A^{k-1}, \pi_B^{k-1})$ such that all jobs $\{1,\ldots,k-1\}$ are feasibly scheduled and the completion time on either machine, say machine A, was minimized leads to an optimal schedule since if J_k could ever be scheduled feasibly, it could be appended to the end of the resulting schedule on machine A.

This optimal rearrangement can be obtained through solving the following problem.

$$\text{minimize} \quad \sum_{j=1}^{k-1} p_j x_{Aj}$$

$$\text{subject to} \quad \sum_{j=1}^{i} p_j x_{Aj} \leq d_i \qquad i = 1,\ldots,k-1$$

$$\sum_{j=1}^{i} p_j x_{Bj} \leq d_i \qquad i = 1,\ldots,k-1 \qquad (7\text{-}4)$$

$$x_{Aj} + x_{Bj} = 1 \qquad j = 1,\ldots,k-1$$

$$x_{Aj},\ x_{Bj} = 0,1 \qquad j = 1,\ldots,k-1.$$

The number of variables in (7-4) can be reduced to k-1 by letting $x_{Bj} = 1 - x_{Aj}$ and rewriting the constraints as

$$\sum_{j=1}^{i} p_j - d_i \leq \sum_{j=1}^{i} p_j x_{Aj} \leq d_i \qquad i = 1,\ldots,k-1 \qquad (7\text{-}5)$$

$$x_{Aj} = 0,1 \qquad j = 1,\ldots,k-1.$$

The resulting algorithm for the two machine problem would be to schedule jobs arbitrarily until an infeasible partial schedule is developed at index k in the due-date sequence. Solve problem (7-4) and attempt to append J_k at the end of the resulting schedule on machine A. If it fits, continue to schedule as before. If it will not fit, exclude

that job currently scheduled on either machine having longest processing time. Continue to schedule as before. Each time an infeasible partial schedule is developed, apply the same exclusion procedure.

What remains to be done to make this algorithm of practical use is the development of an efficient solution method for problem (7-4).

The two machine problem (7-3) is easily generalized to m machines as follows.

$$\text{maximize} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij}$$

$$\text{subject to} \quad \sum_{j=1}^{k} p_j x_{ij} \leq d_k \qquad i = 1,\ldots,m \quad k = 1,\ldots,n \qquad (7\text{-}6)$$

$$\sum_{i=1}^{m} x_{ij} \leq 1 \qquad j = 1,\ldots,n$$

$$x_{ij} = 0,1 \qquad i = 1,\ldots,m \quad k = 1,\ldots,n$$

Contingent upon development of a solution method for problem (7-4) efficient enough for practical use, the algorithm stated for the two machine problem could be generalized to handle this problem.

D. OTHER PROBLEMS

Other interesting problems include one in which the investigator can temporarily increase speed but must pay a price to do so, an example being use of after-burner on

interceptor aircraft. This problem can be translated into a job-shop problem with one machine, n jobs, each with processing time p_i, due-date d_i, and each having value V_i. The operator has the option of running the machine at two speeds - normal, which costs zero dollars, or fast, which costs C dollars per unit time. The problem is to determine that schedule with maximum value.

Another interesting formulation is a further generalization of the two machine problem discussed in Section VII,C where processing times differ from machine to machine, along with the value for processing. None of these formulations are considered here.

LIST OF REFERENCES

1. Balut, S.J., "Scheduling to Minimize the Number of Late Jobs When Set-up and Processing Times are Uncertain," to be published in a future issue of Management Science.

2. Balut, S.J., and Howard, G.T., Minimizing the Number of Penetrations in a Boundary Defense Problem, paper presented at the 40th National Meeting of the Operations Research Society of America, Los Angeles, Calif. 29 October 1971.

3. Barachet, L.L., "Graphic Solution of the Traveling Salesman Problem," Operations Research, V. 5, p. 841-845, 1957.

4. Bellmore, M., and Nemhauser, G.L., "The Traveling Salesman Problem: A Survey," Operations Research, V. 16, p. 538-558, 1968.

5. Brown, A.P.G., and Lomnicki, Z.A., "Some Applications of the 'Branch-and-Bound' Algorithm to the Machine Scheduling Problem," Operational Research Quarterly, v. 17, No. 2, June 1966.

6. Brown, R.G., "Simulations to Explore Alternative Sequencing Rules," Naval Research Logistics Quarterly, v. 15, p. 281-286, June 1968.

7. Busacker, R.G., and Saaty, T.L., Finite Graphs and Networks, McGraw-Hill, 1965.

8. Center for Naval Analysis Memorandum for DOR, NAVWAG, (NWG) 11-611, Investigation Theory, Problems for Consideration by OEG Consultants, by J. A. Neuendorffer, 24 January 1961.

9. Charnes, A., and Cooper, W. W., "Chance-Constrained Programming," Management Science, v. 6, 1959.

10. Charnes, A., and Cooper, W. W., "Deterministic Equivalents for Optimizing and Satisficing under Chance Constraints," Operations Research, v. 11, 1963.

11. Clark, C. E., "The Greatest of a Finite Set of Random Variables," Operations Research, v. 9, No. 2, p. 145-162, March-April 1961.

12. Conway, R.W., Maxwell, W.L., and Miller, L.W., Theory of Scheduling, Addison-Wesley, 1967.

13. Cornell University Dept. of Operations Research Report 51, A Simplified Algorithm for Sequencing n Jobs on One Machine to Minimize the Number of Late Jobs, by H. Emmons, August 1958.

14. Dantzig, G.B., "Linear Programming under Uncertainty," Management Science, v. 1, p. 197-206, 1955.

15. Dantzig, G.B., and Madansky, A., "On the Solution of Two Stage Linear Programs under Uncertainty," Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, v. 1, p. 165-176, 1961.

16. Elmaghraby, S.E., "The Machine Sequencing Problem - Review and Extensions," Naval Research Logistics Quarterly, v. 15, June 1968.

17. Flood, M.M., "The Traveling Salesman Problem," Operations Research, v. 4, p. 61-75, 1956.

18. Ford, L.R. Jr., and Fulkerson, D.R., Flows in Networks, Princeton University Press, 1962.

19. Gavet, J.W., "Three Heuristic Rules for Sequencing Jobs to a Single Production Facility," Management Science, v. 11, No. 8, June 1965.

20. Gere, W.S., "Heuristics in Jobshop Scheduling," Management Science, v. 13, No. 13, November 1966.

21. Held, M., and Karp, R. M., "A Dynamic Programming Approach to Sequencing Problems," SIAM, v. 10, p. 196-210, 1962.

22. Ignall, E., and Schrage, L., "Applications of the Branch and Bound Technique to Some Flowshop Scheduling Problems," Operations Research, v. 13, 1965.

23. Jackson, J.R., "An Extension of Johnson's Results on Job-Lot Scheduling," Naval Research Logistics Quarterly, v. 3, No. 3, September, 1956.

24. Johnson, S.M., "Optimal Two and Three Stage Production Schedules with Set-up Times Included," Naval Research Logistics Quarterly, v. 1, No. 1, March 1954.

25. Larson, H.J., Introduction to Probability Theory and Statistical Inference, p. 178, Wiley, 1969.

26. Lawler, E.L., and Wood, D.E., "Branch and Bound Methods: A Survey," Operations Research, v. 14, p. 699-719, 1966.

27. Little, J.D.C., Murty, K.G., Sweeney, D.W., and Karel, C., "An Algorithm for the Traveling Salesman Problem," Operations Research, v. 11, p. 979-989, 1963.

28. Lomnicki, Z.A., "A 'Branch and Bound' Algorithm for the Exact Solution of the Three-Machine Scheduling Problem," Operational Research Quarterly, v. 16, No. 1, March 1965.

29. Madansky, A., "Methods of Solution of Linear Programs Under Uncertainty," Operations Research, v. 10, p. 463-471, 1962.

30. Maxwell, W.L., "On Sequencing n Jobs on One Machine to Minimize the Number of Late Jobs," Management Science, v. 16, No. 5, January 1970.

31. McMahon, G.B., and Burton, P.G., "Flow-Shop Scheduling with the Branch and Bound Method," Operations Research, v. 15, No. 3, May-June 1967.

32. Moore, J.M., "An n Job, One Machine Sequencing Algorithm for Minimizing the Number of Late Jobs," Management Science, v. 15, No. 1, September 1968.

33. Nemhauser, G.L., Introduction to Dynamic Programming, Wiley, 1966.

34. Neumark, S., Solution of Cubic and Quartic Equations, Pergamon Press, 1965.

35. Operations Evaluation Group Internal Memorandum OEG 656-70, Investigation Theory Survey, Some Problems and Their Proposed Solutions, by S.J. Balut, 23 October 1970.

36. Operations Evaluation Group Internal Memorandum OEG 1013-61, Literature Search Relative to Investigation Theory, by A. Sheerer, 10 August 1961.

37. Saaty, T.L., Optimization in Integers and Related Extremal Problems, p. ix, McGraw-Hill, 1970.

38. Shapiro, D., Algorithms for the Solution of the Optimal Cost Traveling Salesman Problem, Sc.D. Thesis, Washington University, St. Louis, 1966.

39. Strum, L.B.J.M., "A Simple Optimality Proof of Moore's Sequencing Algorithm," Management Science, v. 17, 1970.

40. UCLA Management Science Research Project Report 40, Scheduling a Production Line to Minimize Maximum Tardiness, by J.R. Jackson, January, 1955.

41. U.S. Naval Postgraduate School report NPS55HK72021A, N Job, One Machine Scheduling to Minimize the Number of Late Jobs When Set-Up Times are Sequence Dependent, by S.J. Balut and G.T. Howard, February 1972.

42. U.S. Naval Postgraduate School Report NPS55HK73011A, Comparison of Three Rules of Thumb to the Optimal Solution in Investigation Theory - With Sample Problems, by S.J. Balut and G.T. Howard, January 1973.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Dept. of Operations Research and Administrative Sciences, Code 55 Naval Postgraduate School Monterey, California 93940	1
4.	Chief of Naval Personnel Pers 11b Department of the Navy Washington, D.C. 20370	1
5.	Assoc. Professor G.T. Howard Code 55Hk Dept. of Operations Research and Administrative Sciences Naval Postgraduate School 93940	1
6.	Professor D.E. Harrison Code 61Hx Department of Physics Naval Postgraduate School Monterey, California 93940	1
7.	Assoc. Professor U.R. Kodres Code 53kr Department of Mathematics Naval Postgraduate School Monterey, California 93940	1
8.	Asst. Professor R.H. Shudde Code 55Su Dept. of Operations Research and Admin. Sci. Naval Postgraduate School Monterey, California 93940	1
9.	Asst. Prof. R. W. Butterworth Code 55Bd Dept. of Operations Research and Admin. Sci. Naval Postgraduate School Monterey, California 93940	
10.	LCDR Stephen J. Balut, USN SMC 2574 Naval Postgraduate School Monterey, California 93940	2

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

ORIGINATING ACTIVITY (Corporate author)
Naval Postgraduate School
Monterey, California 93940

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

REPORT TITLE
Problems in Investigation Theory

DESCRIPTIVE NOTES (Type of report and inclusive dates)
Ph.D. Thesis; March, 1973

AUTHOR(S) (First name, middle initial, last name)
Stephen John Balut

REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
March 1973	163	42

CONTRACT OR GRANT NO.

PROJECT NO.

9a. ORIGINATOR'S REPORT NUMBER(S)

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

DISTRIBUTION STATEMENT
Approved for public release; distribution unlimited.

SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY
Naval Postgraduate School
Monterey, California 93940

ABSTRACT

Investigation theory treats discrete combinatorial optimization problems in which there are several objects passing through a region containing one or more investigators who are to investigate, according to some criteria, objects prior to their escape across a portion of the boundary of the region. In general, investigation times are sequence-dependent functions of the time investigation is initiated. This research treats problems with one investigator under the criteria of minimization of the number of objects to escape uninvestigated. Those problems for which optimal solutions can be efficiently obtained are identified and algorithms developed. For the general problem, heuristic solution methods are suggested and evaluated through comparison of results obtained with optimal solutions. An analysis is presented for problems with uncertain investigation times and also for problems in which objects are not immediately available for investigation. Generalizations to more than one investigator and an alternate objective are discussed. The relationship between investigation and jobshop scheduling problems is illustrated throughout.

DD FORM 1 NOV 68 1473 (PAGE 1)
S/N 0101-807-6811

Security Classification

A-31408

KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Investigation Theory						
Boundary Defense						
Sequence-Dependent						
Jobshop Scheduling						
Release-Time Scheduling						
Discrete Optimization						

Thesis 142003
B208 Balut
c.1 Problems in investiga-
tion theory.

thesB208
Problems in investigation theory.
3 2768 001 00676 0
DUDLEY KNOX LIBRARY